Universitat de Lleida

TREBALL FINAL DE GRAU

ESCOLA POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: Ricard Zapater Muñoz

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: **Implementació de la lògica de funcionament d'un dispositiu per automatitzar una cadira de rodes manual per a nens i nenes amb discapacitat**

Director/a: **Antoni Granollers Saltiveri, Miquel Nogués Aymamí**

Presentació

Mes: Juliol

Any: 2019

Índex

Índex de figures

1 Introducció

Aquest treball de fi de grau neix d'un treball anterior, realitzat l'any 2018 per un alumne del grau d'enginyeria mecànica [1], en el qual es va acabar amb el disseny conceptual d'un dispositiu que aniria acoblat a una cadira de rodes per a nens i nenes amb discapacitat de l'Associació AREMI.

Aquest dispositiu serveix per motoritzar una cadira de rodes manual pediàtrica per automatitzar el seu moviment. L'objectiu és que els seus usuaris puguin experimentar el moviment que realitza el dispositiu com a resposta a les accions que ells mateixos fan.

En el present treball de fi de grau s'implementarà la lògica de funcionament per tal que es pugui començar a fer un ús del dispositiu per part de l'Associació AREMI.

1.1 Objectius

Tal com s'ha explicat anteriorment, l'objectiu del present treball de fi de grau és el d'implementar la lògica necessària per desenvolupar el dispositiu que anirà acoblat a una cadira de rodes manual.

El dispositiu tindrà els següents modes de funcionament:

- Mode 1: L'usuari prem un botó i el dispositiu efectua 5 segons de moviment autònom evitant obstacles.
- Mode 2: Mateix mode que l'anterior amb la diferència que només s'efectua mentre s'està prement un botó.
- Mode 3: L'educador de l'Associació AREMI és el que controla el moviment del dispositiu mitjançant un comandament.

Per últim, un objectiu important també és fer que el manteniment del dispositiu sigui senzill, per tal de facilitar les modificacions i/o ampliacions que es vulguin fer en un futur.

Resumint els objectius obtenim els següents punts:

- Implementar la lògica de funcionament necessària per poder utilitzar el dispositiu.
- Desenvolupar els diferents modes de funcionament, cadascun amb les seves característiques.
- Facilitar la continuació del projecte i el manteniment del dispositiu.

1.2 Estructura de la memòria

La memòria s'ha estructurat de la següent forma:

Inicialment s'expliquen els objectius del treball i els seus antecedents.

A continuació, en l'estat de l'art s'explica les diferents tecnologies i components utilitzats per la realització del treball. També s'explica els estudis realitzats per desenvolupar aquest treball.

Després de l'estat de l'art s'explica la part del desenvolupament, la qual és la més extensa. En aquesta part s'expliquen les fases del desenvolupament i les seves dificultats.

Finalment es parla del treball futur d'aquest projecte i les conclusions del treball de fi de grau. També es disposa de la bibliografia consultada per la realització del treball i els annexos.

1.3 Antecedents

Els antecedents són importants per entendre d'on parteix aquest treball de fi de grau. Tal com s'ha explicat anteriorment, l'any 2018 un alumne del grau d'enginyeria mecànica va realitzar el disseny d'un dispositiu i l'estructura que aniria acoblada a la cadira de rodes.

En l'anterior treball es van realitzar diverses reunions entre la UdL i l'Associació AREMI, i es va acabar amb un disseny conceptual del dispositiu.

L'Associació AREMI és una associació sense ànim de lucre situada a Lleida. Aquesta associació té la finalitat de donar suport a persones amb discapacitats de diferents tipus al llarg de totes les etapes de la seva vida, i així poder millorar la seva qualitat de vida.

L'associació disposa de diferents tipus de cadires i un dels objectius del dispositiu és que es pugui utilitzar en 3 tipus de cadires de rodes manuals pediàtriques, les quals es poden observar en la figura 1. A més de poder ser muntat en aquest tipus de cadires, també ha de tenir un muntatge fàcil que permeti ser muntat per una sola persona en menys de 4 minuts.

Tal com es pot observar en la figura 1 tenen estructures diferents, tot i això el dispositiu es pot utilitzar en les diferents estructures. La idea es basa en acoblar el dispositiu de tal forma que s'elevin les rodes fixes i quedin lliures les rodes boges.

En la figura 2a es pot observar com es realitzaria l'acoblament a la cadira *Invacare Action 3 Junior Estàndard*. Tal com es pot observar, s'eleven les dues rodes fixes posteriors i es deixarien lliures les dues rodes boges frontals.

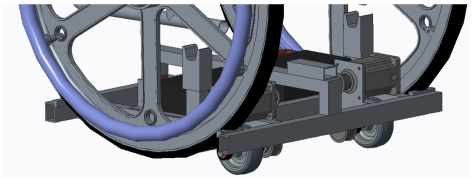
Les cadires *Patron Rehatom 4* i *Ottobock Kimba Neo* tenen una estructura diferent de l'anterior cadira de rodes. En aquestes cadires s'utilitza la barra entre les dues rodes fixes per elevar el dispositiu. En la figura 2b es pot observar com s'elevarien les rodes.

(a) Invacare Action 3 Junior Estàndard

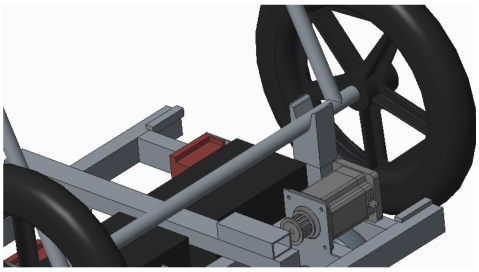
(b) Patron Rehatom 4

(c) Ottobock Kimba Neo

Figura 1: Tipus de cadires de rodes



(a) Acoblament per la cadira Invacare Action 3 Junior Estàndard



(b) Acoblament per les cadires Patron Rehatom 4 i Ottobock Kimba Neo

Figura 2: Acoblaments pels diferents tipus de cadires de rodes

En les figures anteriors es pot observar com tots els elements estan exposats sense cap protecció. En la figura 3 es pot observar el disseny final amb totes les proteccions exteriors. Tots els elements pel funcionament del dispositiu estan situats en l'interior, tret dels sensors ultrasònics, els quals estarien situats en els reposapeus. Més endavant s'explica amb més detall que són els sensors ultrasònics i perquè s'utilitzen.

El present treball està enfocat al desenvolupament de la lògica de funcionament, però també ha tingut una part multidisciplinària, ja que durant el desenvolupament també s'ha hagut d'avaluar aspectes electrònics i mecànics.

S'ha de tenir en compte que durant el desenvolupament s'ha hagut de fer modificacions i addicions al disseny inicial, ja que han anat sorgint aspectes que no es van plantejar inicialment, o es van percebre d'una forma determinada que en la pràctica no ha donat els resultats que s'esperaven i s'ha hagut de buscar una alternativa.

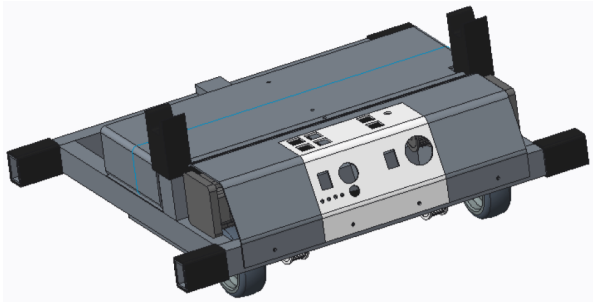


Figura 3: Disseny final amb protecció exterior

A continuació és mostra un resum dels punts més rellevants del disseny conceptual:

- Velocitat i acceleració variable: S'ha de poder regular la velocitat i l'acceleració per adequar-se als diferents usuaris o fins i tot poder regular la velocitat segons l'espai en el qual estigui el dispositiu.
- Modes de conducció: Explicats anteriorment.
- Recorregut programat: El dispositiu ha de poder realitzar diferents recorreguts de forma autònoma per desplaçar-se entre els diferents espais del recinte.

L'últim punt no s'ha pogut implementar, ja que han anat sorgint diferents problemes de funcionament bàsic i s'ha preferit poder assegurar una bona base per facilitar les pròximes addicions i modificacions que es realitzin del dispositiu.

2 Estat de l'art

En aquesta secció s'expliquen les tecnologies i components utilitzats en aquest treball. També s'expliquen els estudis realitzats per començar amb el desenvolupament del dispositiu.

2.1 Tecnologies i components utilitzats

En el treball de fi de grau anterior es van escollir diferents tecnologies i components per a la construcció del dispositiu. A continuació s'explica quins elements es van escollir i el seu objectiu.

Es va escollir la plataforma Arduino per encarregar-se de tota la lògica de funcionament i comunicació amb els components. Es podria dir que és el cervell del dispositiu.

Per moure les rodes del dispositiu s'utilitzen motors pas a pas, els quals a partir del seu moviment s'aconsegueix desplaçar el dispositiu en una direcció amb una determinada velocitat i acceleració.

El dispositiu té dos modes de funcionament autònoms i un de manual. Pels modes autònoms, s'utilitzen sensors ultrasònics per mesurar la distància entre el sensor i un obstacle davant seu, i així poder evitar col·lisions modificant la direcció del dispositiu.

Pel mode de funcionament manual, es va escollir un comandament de la consola PlayStation 2 perquè l'educador de l'Associació AREMI pogués controlar la direcció del dispositiu utilitzant els diferents botons del comandament.

En el treball anterior no es va contemplar, però per la realització de la documentació s'ha utilitzat l'eina Doxygen, la qual especifica una forma de documentar el codi que després es pot exportar a diferents formats.

Cal remarcar que en el disseny es van especificar una sèrie d'elements que no han sigut proporcionats pel prototip, però que s'han d'afegir en la seva construcció. Aquests elements serien els següents:

- Selector de modes, utilitzat per poder canviar el mode de funcionament. Actualment s'ha definit la funció i s'ha provat el seu funcionament de forma alternativa, però no es disposa de cap element físic per realitzar aquest canvi.
- Leds indicatius per indicar quin mode hi ha activat en cada moment.
- Polsador adaptat perquè l'usuari de la cadira pugui interactuar amb el dispositiu.

A continuació s'expliquen amb més detall les característiques més rellevants dels diferents elements.

2.2 Arduino

Arduino és una plataforma open-source per crear projectes d'electrònica. Arduino consisteix en una placa electrònica, normalment anomenada microcontrolador, la qual té la capacitat de llegir *inputs* i convertir-los en *outputs*.

Hi ha diferents tipus de plaques Arduino les quals varien en la capacitat de processament, nombre de pins, mida, connexions... En la secció 2.2.2 s'explica amb més detall cadascuna dels diferents tipus de plaques.

Els *inputs* poden ser dades rebudes de sensors, com per exemple, sensors de proximitat, de temperatura, d'humitat... A més de la informació dels sensors també es pot rebre informació sobre l'estat de diferents botons, potenciòmetres, commutadors, *joysticks*...

Aquests *inputs* poden servir per després realitzar diferents càlculs i convertir-los en *outputs*.

Els *outputs* poden ser de diferents tipus, per exemple, encendre un led, mostrar informació per una pantalla, moure un motor...

Independentment del component que es connecti al microcontrolador, aquest rep i envia informació a través de l'estat del senyal dels seus pins.

Totes les accions s'han de programar en un llenguatge que tècnicament és C/C++, però amb algunes abstraccions. Quan es vol carregar a l'Arduino, es fan tasques de preprocessament per transformar el codi a C++, es compila i es carrega en l'Arduino. Per la comunicació entre l'Arduino i l'ordinador s'utilitza el port sèrie.

Durant la implementació de la lògica de funcionament, es solen utilitzar llibreries per abstreure la gestió dels components de *input* i *output*, tot i que es pot realitzar sense cap llibreria, és recomanable buscar llibreries per tenir un funcionament més fiable. També cal remarcar que de vegades no s'aconsegueix el funcionament esperat i s'ha de realitzar a baix nivell sense cap capa d'abstracció.

L'estructura d'un projecte d'Arduino ve definida per dues funcions:

- *void setup()*: Cada vegada que s'inicialitza l'Arduino, s'executa primer aquesta funció.
- *void loop()*: Una vegada s'ha acabat d'executar el *setup*, es passa a la funció *loop* i es repeteix contínuament com si fos un bucle infinit, tot i que internament té alguns elements de gestió de processador i memòria.

Una altra part important pels projectes d'Arduino són les interrupcions. Les interrupcions interrompen el bucle principal per executar una funció determinada, i quan acaba torna al programa principal. Hi ha dos tipus d'interrupcions:

- Interrupcions per senyal: Aquest tipus d'interrupcions es produeixen quan es produeix un canvi de senyal en un dels pins. Es pot definir quan s'ha de realitzar la interrupció en funció de l'estat del senyal, el qual pot ser un dels següents:
  - *LOW*: Interrupció quan el pin no té senyal.
  - *CHANGE*: Interrupció quan el pin canvia d'estat.
  - *RISING*: Interrupció quan el pin canvia l'estat de *LOW* a *HIGH*.
  - *FALLING*: Interrupció quan el pin canvia l'estat de *HIGH* a *LOW*.
- Interrupcions per temps: Aquest tipus d'interrupció es produeixen cada x microsegons. Es poden aturar i reprendre, i també es pot modificar cada quant temps s'ha de produir.

Un aspecte a tenir en compte és el fet que degut a la forma d'execució, s'ha d'utilitzar en gran part variables globals. Sobretot és un aspecte rellevant quan es tracta de les funcions que criden les interrupcions, ja que aquestes funcions no poden rebre paràmetres d'entrada. Aquestes variables s'ha de marcar com *volatile* per indicar que el valor pot ser modificat fora de la secció de codi on apareix, ja que pot ser modificat quan es produeix una interrupció. També s'indica que aquestes variables no s'han de carregar en un registre i s'han de carregar en RAM per poder tenir un accés més ràpid.

2.2.1 Com executar un programa d'Arduino

Per executar un programa d'Arduino es necessita tenir l'IDE d'Arduino, un cable USB i la placa Arduino.

La plataforma Arduino disposa del seu propi IDE (*Integrated Development Environment*), el qual es tracta d'un IDE senzill que permet desenvolupar en aquesta plataforma i carregar el programa en la placa electrònica.

En la figura 4 s'ha anotat els aspectes més rellevants de l'IDE:

1. *Verify*: Serveix per compilar el codi i comprovar que no es produeix cap error.
2. *Upload*: Es compila el codi, i es carrega en l'Arduino que hi ha connectada.
3. *Serial Monitor*: Obre una finestra per monitorar el port sèrie.
4. Finestra per monitorar el port sèrie on es pot veure la informació que mostra el port i també es pot enviar informació a l'Arduino connectada.

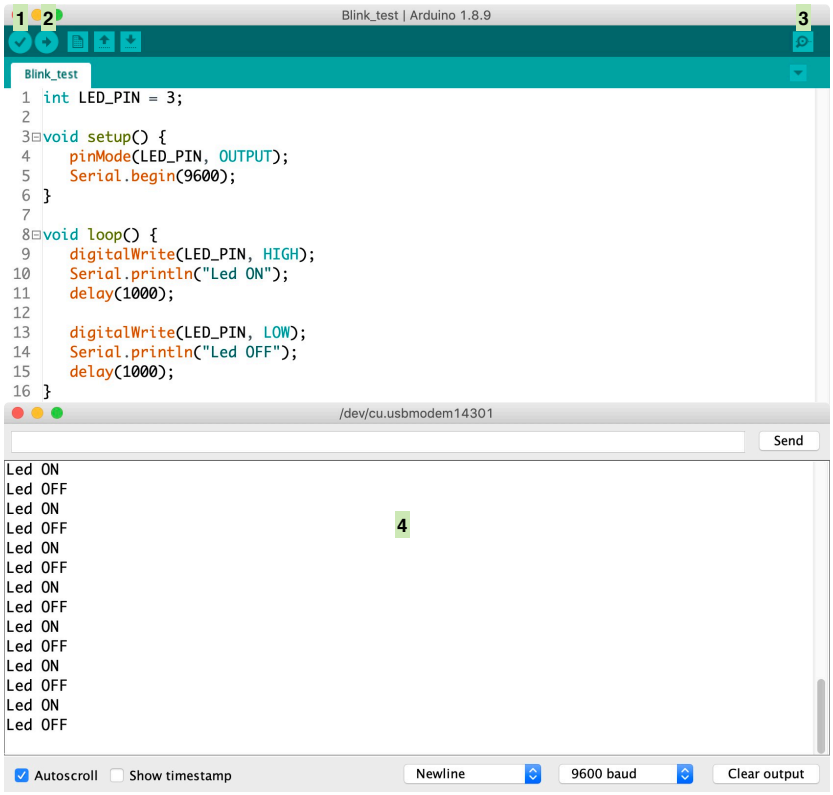


Figura 4: Arduino IDE

L'exemple que es pot veure en la figura 4 és el següent codi, el qual s'encarrega d'encendre i apagar un led cada 1 segon, com si es tractés d'un intermitent en un automòbil.

```
int LED_PIN = 3;

void setup()
{
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  digitalWrite(LED_PIN, HIGH);
  Serial.println("Led ON");
  delay(1000);

  digitalWrite(LED_PIN, LOW);
  Serial.println("Led OFF");
  delay(1000);
}
```

El codi comença executant la funció *setup*. En aquesta funció es defineix el mode del pin del led com a OUTPUT i s'inicialitza la comunicació amb el port sèrie amb una velocitat de 9600 bits/segon (bauds).

Després del *setup*, s'executa contínuament la funció *loop* com si es tractés d'un bucle infinit. En aquesta funció es realitza el següent:

- Encendre led i indicar pel port sèrie que el led està encès.
- Esperar 1 segon.
- Apagar led i indicar pel port sèrie que el led està apagat.
- Esperar 1 segon.

Per controlar el led s'ha de realitzar a baix nivell controlant el senyal del pin del led. Per encendre el led es puja el senyal del seu pin, i per apagar el led es baixa el senyal.

Finalment es connecta l'Arduino a l'ordinador utilitzant un cable USB i es carrega el codi a l'Arduino utilitzant la funcionalitat *Upload* de l'IDE.

En cas que no es pugui carregar o que no funcioni correctament, s'ha de comprovar les connexions i també verificar que l'IDE ha detectat correctament quins tipus de placa s'està utilitzant.

2.2.2 Tipus de plaques Arduino

Hi ha diferents tipus de plaques Arduino, les quals varien en el processador, memòria, ports d'entrada/sortida... La varietat facilita l'elecció d'una placa que s'adapti a les necessitats del projecte que es vulgui desenvolupar. A continuació es realitza una breu descripció de cadascuna d'elles.

Arduino Uno és la placa recomanada per iniciar-se, ja que disposa de tots els elements necessaris per començar a desenvolupar en aquesta plataforma. En la figura 5 es pot observar la placa. Les característiques principals són les següents.

- Processador: 16Mhz ATmega328.
- Memòria: 2KB SRAM, 32KB flash.
- Ports E/S digitals: 14.
- Ports E/S analògics: 6 entrada, 0 sortida.

L'alimentació de la placa electrònica es pot realitzar connectant un cable USB a l'ordinador o mitjançant una font externa. A més també disposa d'un botó en la placa per reiniciar l'execució.

Figura 5: Arduino Uno

Si l'Arduino Uno és massa gran pel projecte, es pot utilitzar l'**Arduino Nano**, la qual té el mateix processador que l'Uno, però té unes dimensions inferiors i no disposa d'alimentació amb una font externa.

Per altra banda també es disposa de la placa **Arduino Mega**, la qual ofereix una gran quantitat de ports d'entrada/sortida i major rendiment que les anteriors, tot i que la seva mida també és superior. En la figura 6 es pot observar l'Arduino Mega i les seves característiques són les següents:

- Processador: 16MHz ATmega2560.
- Memòria: 8KB SRAM, 256KB flash.
- Ports E/S digitals: 54.
- Ports E/S analògics: 16 entrada, 2 sortida.

Figura 6: Arduino Mega

Totes les plaques anteriors treballen amb un processador de 8 bits, però també es disposa de plaques amb processadors de 32 bits, les quals ofereixen un millor rendiment, però tenen un consum major.

La placa **Arduino Zero** té el mateix només de pins que l'Arduino Uno però té un processador de 32 bits. També cal remarcar que aquesta placa utilitza un voltatge de 3.3V a diferència de la majoria de plaques que utilitzen 5V.

Igual que en l'Arduino Uno, aquesta placa també disposa d'adaptacions en unes dimensions més reduïdes.

A més d'aquestes plaques, també es disposa d'altres tipus de plaques que porten incorporades funcionalitats addicionals com ranura per a targetes MicroSD, connexió WiFi o Ethernet.

Cal remarcar que aquestes funcionalitats també es poden afegir a les altres plaques connectant-hi els components corresponents.

2.3 Motor pas a pas

Un motor pas a pas es tracta d'un motor síncron que converteix una sèrie de polsos elèctrics en un desplaçament d'un nombre determinat de graus. Cada desplaçament es considera un pas, i tal com el nom del motor indica, aquest es desplaça pas a pas.

Aquests motors es solen utilitzar en aplicacions on la precisió és important, ja que es pot desplaçar el motor fins a una posició determinada.

En la figura 7 es pot observar un motor pas a pas.

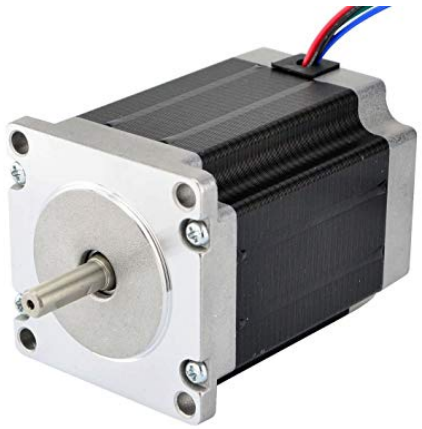


Figura 7: Motor pas a pas

2.4 Sensor ultrasònic

Un sensor ultrasònic serveix per mesurar distàncies utilitzant ones ultrasòniques. El sensor envia una ona ultrasònica que rebota en una superfície i retorna al sensor. A partir del temps d'emissió i recepció es pot calcular la distància entre el sensor i la superfície. En la figura 8 es pot observar un sensor ultrasònic.

Figura 8: Sensor ultrasònic

Una característica important dels sensors ultrasònics és el fet que poden detectar superfícies transparents, ja que es treballa amb ones ultrasòniques. A més també permet detectar correctament superfícies amb pols.

2.5 Comandament PS2

El comandament de la consola PlayStation 2, abreviada com PS2, també serveix per poder comunicar-se amb altres dispositius a més de la consola. En la figura 9 es pot observar el comandament.

Figura 9: Comandament de la consola PlayStation 2

Aquest comandament disposa de diversos botons, palanques i *joysticks*, els quals també es poden pressionar com si es tractés d'un botó.

El comandament envia els senyals de forma analògica, és a dir, es pot determinar la força amb la qual es prem un dels determinats botons.

2.6 Elecció dels components

Tal com s'ha explicat anteriorment, en el treball de fi de grau anterior es va realitzar diverses reunions amb l'Associació AREMI, i a partir de les seves necessitats es van escollir els components més adequats per poder construir el dispositiu.

Abans d'escollir el tipus de placa Arduino, es va analitzar quins components es necessitarien i quin nombre de pins serien necessaris per poder connectar tots els components. Es va considerar que es necessitarien uns 31 pins digitals i es va escollir la placa Arduino Mega, ja que disposa de 54 pins digitals dels quals 23 es deixarien lliures per a futures ampliacions.

Es va considerar utilitzar motors pas a pas perquè ofereixen gran precisió per controlar el seu moviment. Cal tenir present que el dispositiu controla una cadira de rodes que porta una persona a sobre, i per tant és important tenir un comportament precís i realitzar els moviments de forma suau.

Per mesurar distàncies es va proposar utilitzar sensors làser o sensors ultrasònics, però es van descartar els sensors làser perquè no detecten correctament superfícies transparents i són sensibles a la llum solar, i per tant podrien donar problemes en l'exterior. Un altre aspecte és el fet que els sensors làser són més cars que els sensors ultrasònics.

El comandament de PS2 està dissenyat per jugar a videojocs, però els comandaments de les consoles estan dissenyats per ser ergonòmics i facilitar la interacció persona ordinador.

Els comandaments de consola estan començant a ser utilitzats en altres sectors i per altres finalitats que la de jugar a videojocs. Per exemple, actualment s'està utilitzant un comandament de consola per controlar un software de visualització en 3 dimensions d'anatomia humana.

El software s'anomena *BodyViz* [2] i permet visualitzar anatomia humana en 3 dimensions a partir de dades extretes d'Imatges per Ressonància Magnètica (IRM) i Tomografia Computada (TC). El CEO de *BodyViz*, Curt Carlson, explicava que la utilització de teclat i ratolí per controlar la visualització era complicada i poc intuïtiva, però amb el comandament els cirurgians que es preparaven per realitzar una operació invasiva, aconseguien tenir un control més fluid i precís per poder observar l'anatomia virtual del pacient [3].

2.7 Doxygen

Doxygen [4] és una eina que serveix per generar documentació a partir de comentaris en el codi font. Es pot utilitzar en gran part dels llenguatges més utilitzats, com per exemple: C, C++, C#, Java, Python...

El codi es documenta utilitzant blocs de comentaris especials per indicar a Doxygen que es tracta d'un comentari que es considera documentació. En la documentació de Doxygen s'especifica les diferents formes de documentar segons el tipus de llenguatge [5].

En aquests comentaris s'hi afegeixen comandes per especificar diferents característiques, per exemple:

- Autor.
- Versió.
- Breu descripció d'una funció.
- Paràmetres d'una funció.
- Element que retorna una funció.

Aquesta eina té un alt nivell de configuració, els aspectes més rellevants que permet configurar són els següents:

- Quins fitxers s'han d'analitzar o quins fitxers s'han d'excloure.
- En quin format s'ha de generar la documentació i característiques d'aquest.
- Generació grafs i diagrames per representar jerarquia de classes.

La documentació es pot generar en diferents formats:

- HTML.
- LaTeX.
- RTF.
- XML.
- Man page.
- DocBook.

Doxygen està sota la llicència *GNU General Public License*, la qual dóna permís per utilitzar, copiar, modificar i distribuir el software. Tot i això, els documents generats per *Doxygen* són obres derivades dels fitxers que s'han utilitzat per generar aquests documents, i per tant no són afectats per aquesta llicència.

2.7.1 Com utilitzar Doxygen

Per utilitzar aquesta eina s'ha de seguir uns determinats passos, a continuació s'explica el procés que s'ha de dur a terme. S'ha creat un exemple senzill per mostrar el funcionament bàsic. Aquest codi d'exemple simplement rep dos enters i mostra la seva suma.

Primer s'ha de documentar el codi utilitzant comentaris especials per tal que Doxygen ho reconegui com a documentació. En cas que un comentari no segueixi aquesta sintaxi, serà ignorat.

```c
/**
 * @file main.c
 * @author Ricard Zapater
 * @brief Minimal working example to show how Doxygen works
 * @version 1.0
 */
#include <stdio.h>

/**
 * @brief Return the sum of 2 integers.
 *
 * @param num1 First integer
 * @param num2 Second integer
 * @return int The sum of the 2 parameters.
 */
int sum(int num1, int num2)
{
    return num1 + num2;
}

/**
 * @brief Main function of the program. Enter 2 integers and print the sum.
 *
 * @return int Return 0 to indicate that there wasn't any error.
 */
int main()
{
    int num1 = 0;
    int num2 = 0;

    printf("Input the first integer:\n");
    scanf("%d", &num1);

    printf("Input the second integer:\n");
    scanf("%d", &num2);

    printf("The sum is: %d\n", sum(num1, num2));
    return 0;
}
```

A continuació, s'executa la següent comanda en la terminal per crear un fitxer de configuració, el qual se l'ha anomenat *Doxyfile*. Tal com s'ha explicat anteriorment, aquest fitxer és altament configurable.

```
doxygen -g Doxyfile
```

Finalment, per generar la documentació s'executa la següent comanda fent referència al fitxer de configuració:

```
doxygen Doxyfile
```

Cada vegada que es modifiquen els comentaris que formen part de la documentació, s'ha de tornar a executar aquesta última comanda per tornar a generar la documentació.

2.8 Estudis realitzats per desenvolupar aquest treball

Inicialment es va haver d'entendre d'on provenia aquest treball, entendre quines característiques havia de tenir el dispositiu i com havien de funcionar els diferents modes de funcionament.

Per realitzar el desenvolupament s'ha hagut de començar des de zero, ja que no s'havia desenvolupat mai en la plataforma Arduino, ni es coneixia el funcionament dels diferents components que són necessaris per al funcionament del dispositiu. A més, hi havia aspectes electrònics que es desconeixien, o els pocs coneixements que es tenien eren molt bàsics.

Inicialment hi va haver un procés de familiarització amb aquesta plataforma i els diferents components, però durant el desenvolupament s'ha hagut d'anar aprenent diferents conceptes per poder resoldre els problemes que han anat sorgint.

En la pròxima secció s'explicarà el desenvolupament, el qual degut al gran desconeixement que es tenia, ha sigut en gran part de prova i error, sobretot hi ha hagut proves que teòricament semblaven coherents però que després en la pràctica el resultat no era l'esperat.

3 Desenvolupament

El desenvolupament es va dividir en fases que acabarien amb l'obtenció del producte final. Cal tenir en compte que el desenvolupament es va realitzar sense tenir un prototip funcional, és a dir, que els components no estaven acoblats en cap estructura i per tant es va realitzar de forma que fos senzill modificar el funcionament, ja que les accions que s'acabarien realitzant finalment dependrien de com reaccionés el dispositiu en la pràctica.

En la figura 10 es pot observar l'estat del dispositiu. S'han remarcat els components més rellevants:

1. Arduino Mega.
2. Sensors ultrasònics.
3. Motors pas a pas.
4. Comandament PS2.

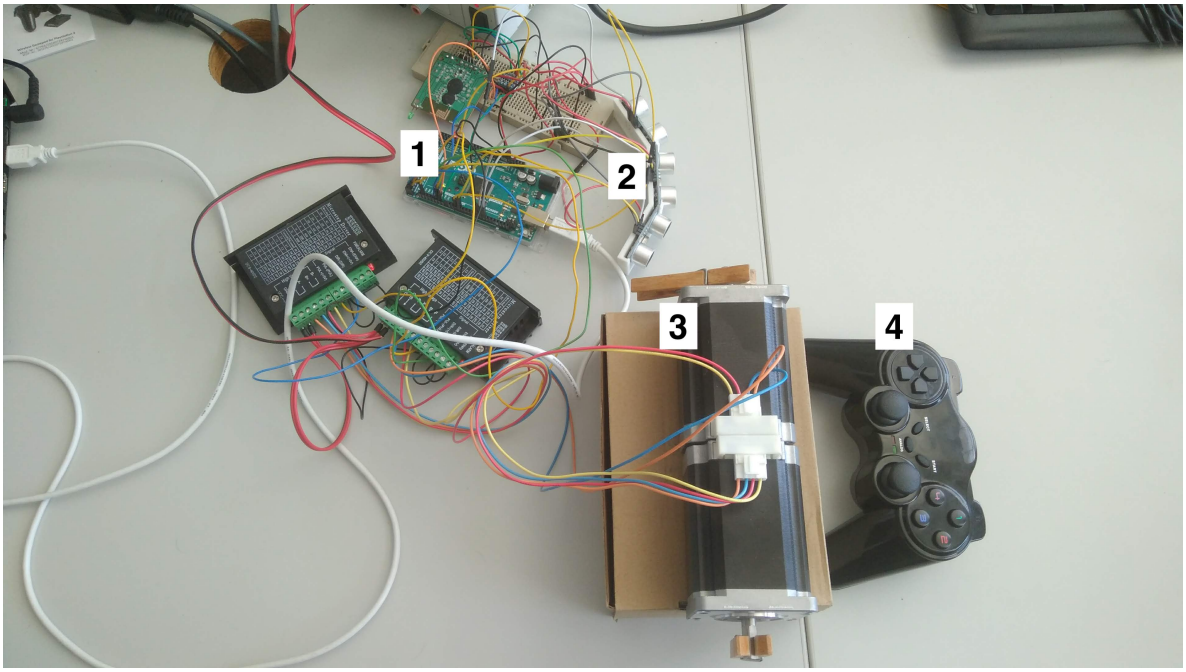


Figura 10: Estat del dispositiu

Tal com s'ha comentat en l'estat de l'art, no es disposa en aquest prototip d'una sèrie d'elements que es van especificar en el disseny, tot i això, haurien d'estar disponibles en la construcció del dispositiu.

3.1 Fases del desenvolupament

Les fases del desenvolupament es van separar segons els components.

Inicialment es va començar pel control dels motors pas a pas, ja que és la base de tot el funcionament del dispositiu.

Per combinar el control dels motors amb els altres components, primer es van realitzar proves aïllades per entendre el funcionament dels components, i a continuació es van combinar amb el control dels motors.

Després de comprovar que la combinació dels components funcionés correctament, es va separar en diferents modes de funcionament, els quals es van dissenyar amb l'objectiu de tenir una estructura unificada per gestionar-los i també facilitar l'addició de nous modes.

Les fases són les següents:

1. Control dels motors pas a pas.
2. Combinar el funcionament dels sensors ultrasònics i els motors.
3. Combinar el funcionament del comandament de PS2 i els motors.
4. Separar-ho en diferents modes de funcionament.

3.2 Motors pas a pas

El control dels motors pas a pas és la base sobre la qual es sustenta tot el funcionament del dispositiu, per tant és la primera fase del desenvolupament. Aquests motors s'encarregaran de moure les rodes del dispositiu i és important tenir un funcionament precís, ja que s'ha de tenir present que a sobre de la cadira de rodes hi haurà un nen/nena que degut a la seva discapacitat pot tenir poca mobilitat o dificultats per respondre ràpidament a imprevistos.

Tot i que, s'ha procurat tenir un funcionament precís i segur, ha d'haver-hi un educador de l'Associació AREMI mentre s'està utilitzant el dispositiu per evitar qualsevol imprevist.

Per controlar els motors pas a pas es va utilitzar la llibreria *AccelStepper*, ja que permet controlar l'acceleració i desacceleració dels motors. Totes les llibreries utilitzades durant el desenvolupament són explicades amb més detall en la secció 3.7. Concretament la llibreria *AccelStepper* es pot consultar en la secció 3.7.1.

Inicialment es va començar amb un motor i es van assolir les següents fites:

1. Moure el motor pas a pas un nombre determinat de passos.
2. Controlar la direcció de gir. (Sentit horari i sentit antihorari)

3. Accelerar i desaccelerar.

Després d'assolir aquestes fites es va afegir un segon motor i es van realitzar proves per poder controlar independentment cadascun dels motors. En aquesta part era important que no es bloquegés l'Arduino quan es donava una ordre a un motor, ja que si no l'altre motor no es mouria fins que aquest hagués arribat a una determinada posició.

Per entendre millor aquest cas i el control dels motors pas a pas, s'ha desenvolupat un exemple senzill però funcional per mostrar el seu funcionament i posteriorment s'ha explicat concretament cada part del codi d'exemple. El funcionament general és: Quan es rep pel port sèrie el caràcter '*f*', llavors el dispositiu avança en línia recta i quan rep el caràcter '*b*' el dispositiu retrocedeix.

```
#include <AccelStepper.h>

// Pins for the right stepper motor
#define RIGHT_STEPPER_STEP_PIN 22
#define RIGHT_STEPPER_DIRECTION_PIN 23

// Pins for the left stepper motor
#define LEFT_STEPPER_STEP_PIN 24
#define LEFT_STEPPER_DIRECTION_PIN 25

int max_speed = 300;           // steps/second
int acceleration = 100;        // steps/second
int one_rev = 200;             // steps

AccelStepper right_stepper(AccelStepper::DRIVER, RIGHT_STEPPER_STEP_PIN,
    RIGHT_STEPPER_DIRECTION_PIN);
AccelStepper left_stepper(AccelStepper::DRIVER, LEFT_STEPPER_STEP_PIN,
    LEFT_STEPPER_DIRECTION_PIN);

void setup()
{
  Serial.begin(9600);

  // Left Stepper
  left_stepper.setMaxSpeed(max_speed);
  left_stepper.setAcceleration(acceleration);

  // Right Stepper
  right_stepper.setMaxSpeed(max_speed);
  right_stepper.setAcceleration(acceleration);
}

void loop()
{
  check_serial();

  right_stepper.run();
  left_stepper.run();
}

```

```
39 void check_serial()
40 {
41   if (Serial.available())
42   {
43     switch (Serial.read())
44     {
45       case 'f':
46         Serial.println("FORWARD");
47         right_stepper.move(one_rev);
48         left_stepper.move(-one_rev);
49         break;
50 
51       case 'b':
52         Serial.println("BACKWARD");
53         right_stepper.move(-one_rev);
54         left_stepper.move(one_rev);
55         break;
56 
57       default:
58         break;
59     }
60   }
61 }
```

Inicialment en la funció *setup* es configuren els motors indicant els seus pins corresponents per indicar els passos i la direcció de gir. També es configura la velocitat màxima i l'acceleració en passos/segon.

La funció *loop* s'executa com si es tractés d'un bucle infinit, i s'encarrega inicialment de cridar a la funció *check_serial*, la qual comprova si ha rebut el caràcter '*f*' o '*b*' pel port sèrie, i s'encarrega d'especificar els passos per moure els motors de tal forma que el dispositiu es mogui endavant o enrere. Més endavant s'explica quines direccions s'han definit i quines accions han de realitzar els motors.

Després de comprovar el port sèrie i especificar els passos, es crida a la funció *run* dels motors. Aquesta funció comprova si el motor té passos pendents de realitzar, però no els realitza tots de cop, sinó que calcula el nombre de passos que ha de realitzar en aquell instant segons el nombre total de passos pendents, l'acceleració i la velocitat, llavors quan es torna a cridar, realitza el següent nombre de passos, per tant aquesta funció ha d'estar en el bucle principal, ja que s'ha de cridar de forma periòdica. Com que aquesta funció no es bloqueja fins que realitza tots els passos, es pot continuar realitzant altres càlculs mentre els motors estan en moviment.

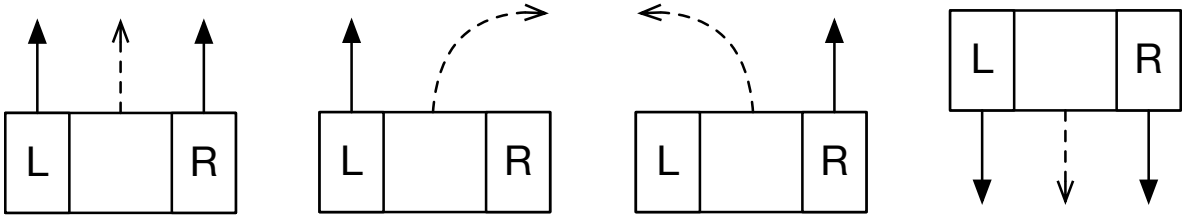
Si s'utilitzés la funció *runToPosition* en comptes de *run*, es bloquejaria el funcionament. En aquest cas, primer es mouria el motor dret fins a arribar al seu destí i a continuació es mouria el motor esquerre fins a arribar al seu, és a dir, no es mourien els dos de forma concurrent.

Per indicar els passos a realitzar s'utilitza la funció *move*, en la qual s'indiquen els passos de forma relativa a la posició actual. Els passos en positiu indiquen que la direcció de gir és en sentit horari i en negatiu indiquen que és en sentit antihorari.

A continuació, s'expliquen les diferents direccions definides per moure el dispositiu i quines accions han de realitzar els motors. Tal com es pot observar en la figura 11, s'han definit 4 direccions:

- FORWARD: El dispositiu avança en línia recta movent el motor esquerre en sentit antihorari i el dret en sentit horari.
- RIGHT: El dispositiu es mou cap a la dreta deixant immòbil el motor dret i movent l'esquerre en sentit antihorari.
- LEFT: El dispositiu es mou cap a l'esquerra deixant immòbil el motor esquerre i movent el dret en sentit horari.
- BACKWARD: El dispositiu es mou enrere movent el motor esquerre en sentit horari i el dret en sentit antihorari.

Quan el dispositiu es mou cap a l'esquerra o la dreta, ho realitza com si es tractés d'un compàs, ja que deixa un extrem immòbil mentre l'altre efectua el moviment. Es va plantejar la idea de fer la rotació sobre l'eix del dispositiu, però es va escollir realitzar el moviment en forma de compàs, ja que es tracta d'un moviment més suau per l'usuari.



(a) FORWARD (b) RIGHT (c) LEFT (d) BACKWARD

Figura 11: Direccions definides

Els diferents modes de funcionament s'encarregaran de decidir quina direcció ha de prendre el dispositiu segons la informació que hagin rebut dels diferents components.

Per exemple, si els sensors ultrasònics detecten que una distància no és segura, s'haurà d'indicar als motors que realitzin un canvi de direcció per tal d'evitar un obstacle. En el cas del comandament de PS2 quan detectés que s'ha premut un botó, indicaria als motors que s'han de dirigir cap a la direcció especificada amb el comandament.

En les pròximes seccions s'explica amb més detall com es combina el control dels motors amb els diferents components.

3.3 Sensors ultrasònics

Els sensors ultrasònics s'utilitzen per mesurar distàncies entre el sensor i una superfície davant seu. La seva funcionalitat en el dispositiu és la de mesurar distàncies per realitzar moviment de forma autònoma evitant obstacles. Amb l'ajuda d'aquests sensors es pot comprovar si la direcció del dispositiu és segura o si ha de fer un canvi de direcció per evitar una col·lisió.

Inicialment es va començar realitzant proves amb un sol sensor ultrasònic per poder veure el seu funcionament i quines limitacions tenia.

Es va començar utilitzant la llibreria *NewPing* per obtenir les distàncies, ja que oferia una forma senzilla de poder rebre les distàncies del sensor i era la llibreria recomanada generalment per treballar amb aquests tipus de sensors. En la secció 3.7.2 s'expliquen amb més detall les seves característiques.

La utilització d'aquesta llibreria era força senzilla i obtenia resultats força precisos, però mentre s'estava fent proves es va observar com els sensors no oferien resultats correctes quan l'ona ultrasònica rebotava en una superfície amb un angle de 30º.

Per evitar aquest cas, es van proposar diverses ubicacions per als sensors, les quals es poden observar en la figura 12.

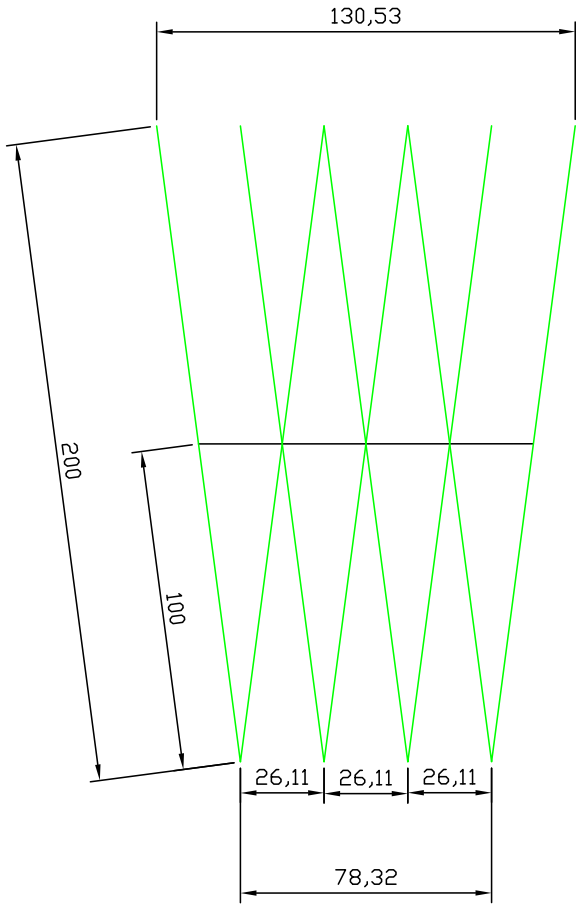
En la figura 12a es pot observar com es disposa de 4 sensors que estan situats tots horitzontalment. En aquesta proposta hi ha el problema que si l'ona ultrasònica rebota en una superfície que té un angle superior a 30º, no hi haurà cap sensor que detecti correctament l'obstacle, ja que tots estan apuntant en la mateixa direcció.

En la figura 12b disposem d'una proposta on hi ha 4 sensors situats de forma vertical, un a sobre de l'altre. En aquest cas tenim més possibilitats de detectar una superfície que tingui un angle superior a 30º. Tot i això, finalment es va decidir utilitzar un disseny que estava pensat específicament per detectar correctament superfícies amb aquesta inclinació.

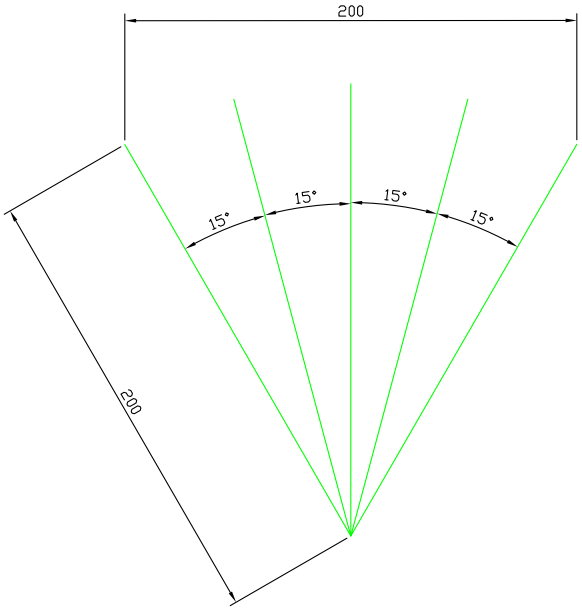
El disseny es va trobar en el blog d'Enrico Formenti [6], i es pot observar en la figura 12c. La idea es basa en situar els sensors en un angle de 30º entre ells, ja que si un sensor té un obstacle amb aquesta inclinació, llavors no obtindrà els resultats correctes, però el sensor que està a 30º d'aquest sí que ho farà. Tal com s'ha explicat, es va escollir aquest disseny perquè està més enfocat a detectar i evitar el cas problemàtic d'un obstacle amb una inclinació superior a 30º.

A partir d'aquest disseny, es va realitzar una figura amb una impressora 3D per poder realitzar les proves, es van afegir els altres dos sensors i es va comprovar de forma aïllada que tots 3 funcionaven correctament. En la figura 13 es poden observar els sensors acoblats a la figura realitzada amb una impressora 3D.

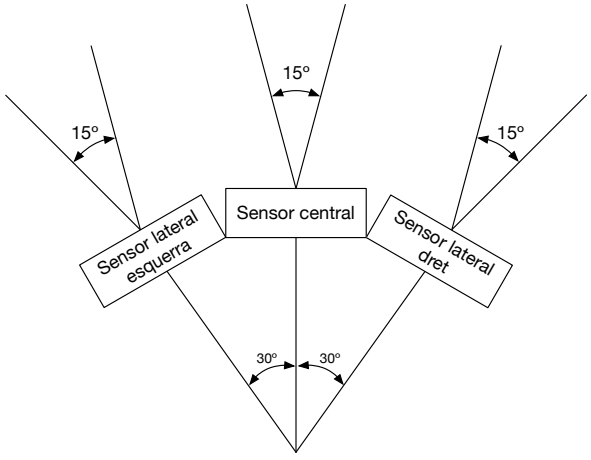
Després de comprovar el funcionament per separat dels sensors ultrasònics i els motors pas a pas, es va combinar el seu funcionament.



(a) Ubicació dels sensors en horitzontal



(b) Ubicació dels sensors en vertical



(c) Ubicació dels sensors en 30º

Figura 12: Propostes per ubicar els sensors

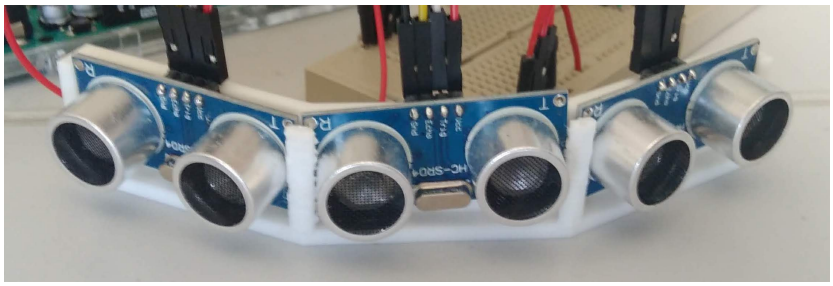


Figura 13: Sensors ultrasònics acoblats a la figura realitzada amb una impressora 3D

3.3.1 Combinar funcionament dels sensors ultrasònics i motors pas a pas

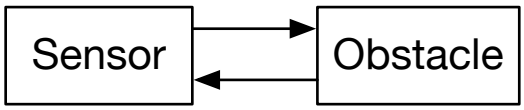
Aquesta fase va comportar un gran nombre de dificultats perquè els dos components funcionessin correctament.

Inicialment es va començar per una prova senzilla en la qual només estava implicat un sensor.

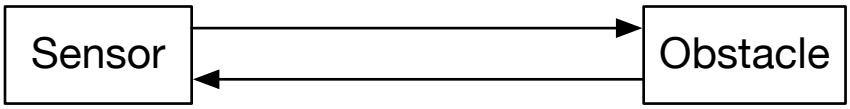
En aquesta prova el dispositiu es movia en línia recta i quan el sensor ultrasònic detectava una distància inferior a un nombre determinat de centímetres, llavors havia de disminuir la velocitat, i quan l'obstacle deixava d'estar present, tornava a la velocitat que duia anteriorment.

Aquesta prova, que a simple vista sembla senzilla, va donar un problema realment interessant. Tal com s'anava apropant l'obstacle al sensor, els motors es movien més ràpidament, i com més lluny estava l'obstacle més lent es movia, és a dir, just al revés del que es volia fer.

En la figura 14 es pot observar una representació del comportament que tenien els motors en aquesta prova.



(a) Motor ràpid



(b) Motor lent

Figura 14: Representació del comportament erroni

Evidentment, el primer que es va comprovar era si el codi era coherent, i teòricament ho era, però en la pràctica el resultat no era l'esperat.

Finalment es va observar que el causant era la llibreria *NewPing* que s'estava utilitzant per mesurar distàncies. Aquesta llibreria quan envia una ona ultrasònica, bloqueja el funcionament i els motors no es poden moure fins que retorna l'ona o es sobrepassa un temps d'espera.

Com es pot observar en la figura 14b, si l'obstacle està lluny, l'ona ultrasònica tardarà més a retornar al sensor, en canvi si té un obstacle proper, figura 14a, l'ona ultrasònica retornarà ràpidament al sensor. Com més petit sigui el temps d'espera, més ràpid es podrà indicar als motors els passos a fer.

És un projecte on el desenvolupament està fortament lligat al món físic, ja que els components del dispositiu tenen les seves limitacions i peculiaritats. En aquest cas, quan s'obtenen distàncies amb els sensors ultrasònics, s'està enviant una ona física que ha de transmetre's per l'aire, rebotar en una superfície i retornar al sensor. Sobretot és un projecte on també s'ha de comprendre correctament el funcionament dels diferents components que hi intervenen per poder desenvolupar una solució funcional.

En aquest moment es va veure que s'havia de realitzar de forma concurrent, ja que de forma seqüencial no funcionava com s'esperava. Vegem a continuació totes les fases per les quals es va passar per poder utilitzar els sensors ultrasònics i els motors pas a pas de forma concurrent.

3.3.2 Sensors ultrasònics de forma concurrent

Fins ara s'havia realitzat de forma seqüencial i per realitzar-ho de forma concurrent era un paradigma diferent.

Inicialment es va investigar com utilitzar fils d'execució, però Arduino no disposa de fils d'execució.

Una altra opció que es va descartar inicialment va ser la d'utilitzar una segona Arduino per aquesta tasca, ja que s'afegia un altre grau de complexitat que es podia evitar si es realitzava només amb una Arduino.

Arribats en aquest punt es va decidir provar-ho de dues formes per veure quina oferia el millor resultat, ja que no es sabia molt bé com reaccionaria en la pràctica.

Forma alternativa d'utilitzar la llibreria *NewPing*

En la documentació de la llibreria *NewPing* es va trobar una forma alternativa d'utilitzar aquesta llibreria [7].

L'autor d'aquesta llibreria va realitzar un exemple en el qual mostrava com es podia utilitzar de tal forma que no bloquegés el funcionament de l'Arduino, i es pogués realitzar altres operacions mentre no es rebia el retorn de l'ona.

El seu funcionament és el següent: Enviar l'ona, i a continuació comprovar cada 24 microsegons si ha rebut el seu retorn, en cas que no l'hagi rebut, continuar amb l'execució.

Es va adaptar l'exemple al nostre cas i es va poder veure com el funcionament dels sensors ultrasònics no afectava el funcionament dels motors pas a pas. Tot i això, cal tenir en compte

que amb aquesta opció s'està utilitzant constantment una part important de la capacitat de processament del microcontrolador i s'ha de recordar que té una capacitat limitada.

Utilitzar interrupcions per hardware

L'altra opció que es va provar va ser la d'utilitzar interrupcions per hardware. En aquesta forma no s'utilitza cap llibreria per obtenir les distàncies i es calcula tot manualment. Per entendre el funcionament amb més facilitat, es defineixen els següents termes:

- Trigger: Moment en el qual es llença l'ona ultrasònica. Quan s'executa el Trigger s'emmagatzema el temps en el qual s'ha executat, i a continuació, puja i baixa el senyal del pin en el qual està connectat el *trigger* del sensor ultrasònic.
- Echo: Moment en el qual es rep l'ona ultrasònica. Quan s'envia l'ona puja el senyal del pin on està connectat l'*echo* del sensor, i la interrupció es produeix quan baixa el senyal, ja que és quan ha retornat l'ona ultrasònica. En aquesta part s'emmagatzema el temps i es realitzen els càlculs necessaris.

En aquesta opció quan s'iniciava l'Arduino, s'executava el Trigger dels 3 sensors a la vegada, i llavors en l'Echo de cada sensor es realitzava el càlcul de la distància i s'executava el Trigger d'aquell sensor.

La idea es basava en el fet que els 3 sensors treballessin concurrentment, però en la pràctica els resultats no eren del tot correctes. El sensor frontal donava resultats correctes, però els laterals donaven resultats completament erronis.

Es va provar individualment cada sensor per detectar on estava l'error, però individualment funcionaven tots correctament, el problema estava quan s'executaven tots a la vegada.

Arribats en aquest punt es va haver de replantejar la forma d'utilitzar els 3 sensors ultrasònics i finalment es va decidir utilitzar una combinació d'interrupcions per hardware i per temps.

Utilitzar interrupcions per hardware i per temps

En la documentació del sensor ultrasònic es recomanava realitzar cicles de mesura de 60 mil·lisegons per evitar interferències entre els diferents senyals [8].

Per poder seguir aquesta recomanació es va afegir una interrupció per temps que es realitzava cada 60ms i executava el Trigger d'un sensor cada 60ms. L'Echo es continuava realitzant amb una interrupció per hardware però en la pujada de senyal s'emmagatzemava el temps d'inici, i en la baixada el de final, i no es realitzava cap càlcul.

Els càlculs es realitzaven una vegada s'hagués passat per tots els sensors. En la figura 15 es pot observar l'esquema de funcionament.

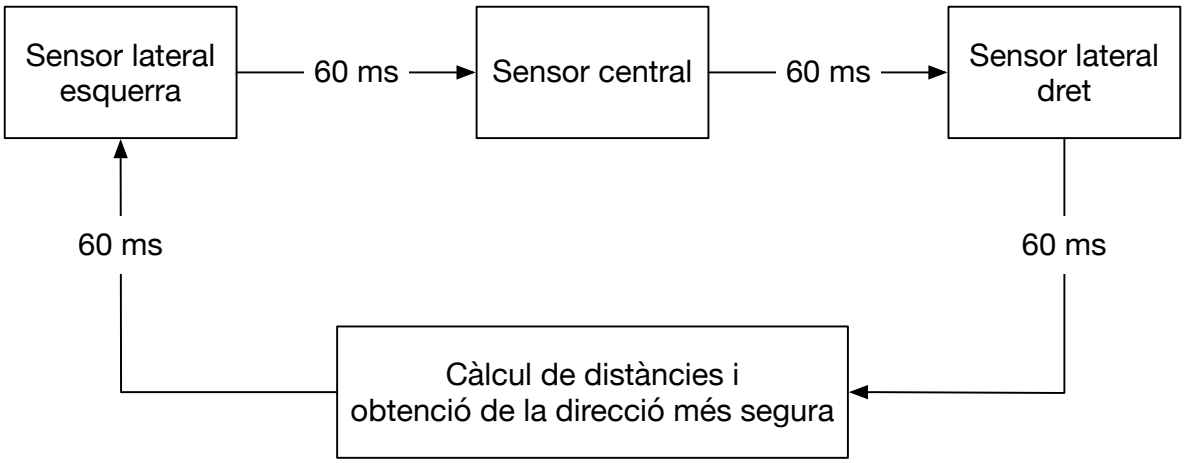


Figura 15: Esquema de funcionament dels sensors ultrasònics

Tal com es pot observar amb 240ms es realitza el cicle complet, per tant en 1 segon es realitzen 4 cicles.

Els càlculs s'han separat de l'Echo, ja que es recomana que les interrupcions siguin el més ràpid possible i per tant es realitzen després d'obtenir tots els temps de tots els sensors.

Finalment, amb aquesta forma s'ha aconseguit el resultat que s'esperava, és a dir, poder utilitzar els sensors ultrasònics i els motors pas a pas concurrentment.

Després de calcular les distàncies de tots els sensors, s'ha de decidir en quina direcció s'ha de moure el dispositiu, i per decidir-ho s'ha realitzat següent algorisme de moviment autònom.

3.4 Algorisme de moviment autònom

L'algorisme de moviment autònom està basat en escollir sempre la direcció més segura i així evitar qualsevol col·lisió.

Per determinar si una direcció és segura o no s'ha especificat un paràmetre que indica amb quants centímetres una distància és considerada segura. Aquest paràmetre es pot modificar fàcilment, ja que s'ha realitzat de forma que sigui senzill modificar el comportament del dispositiu.

En la figura 16 es pot observar un diagrama de flux on es representa l'algorisme.

L'algorisme inicialment comprova si la distància central és segura, en cas que no ho sigui decideix si és més segur girar a l'esquerra o a la dreta.

Si la distància central és segura, es comprova si també ho és la distància que s'ha obtingut del sensor lateral esquerra, i si no ho és, llavors gira a la dreta.

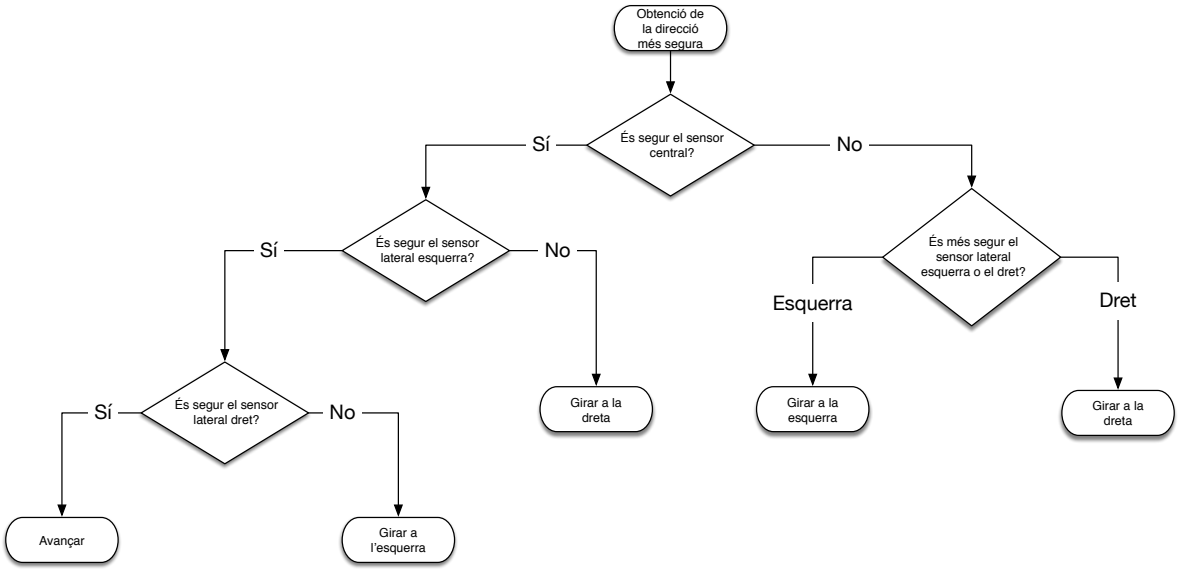


Figura 16: Algorisme de moviment autònom

En cas que la distància central i la lateral esquerra siguin segures, llavors només queda comprovar la distància lateral dreta, i en cas que no sigui segura llavors girar a l'esquerra.

Tal com es pot comprovar, es basa en detectar distancies no segures, però si és dona el cas que totes les distàncies són segures llavors el dispositiu pot continuar avançant en línia recta.

Es tracta d'un algorisme senzill, el qual malgrat no poder haver-lo provat en un prototip funcional, s'han realitzat diferents proves i s'han intuït futures dificultats.

Un aspecte que cal remarcar és el fet que no es retrocedeix mai, ja que només es disposen de sensors frontals i no hi ha sensors en la part posterior. En cas de retrocedir, es podria estar dirigint el dispositiu en una direcció que no fos segura, ja que es desconeix quins obstacles hi poden haver.

En la figura 17 es pot observar un cas en el qual seria útil disposar de sensors posteriors. En aquest cas el sensor central és segur, però els laterals no ho són. Si es segueix l'algorisme de la figura 16 s'escolliria girar a la dreta, tot i que no seria una direcció del tot segura.

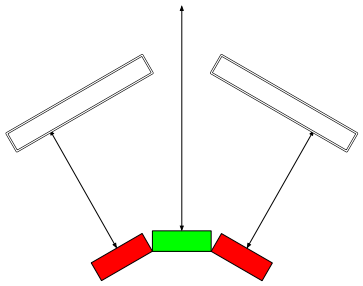
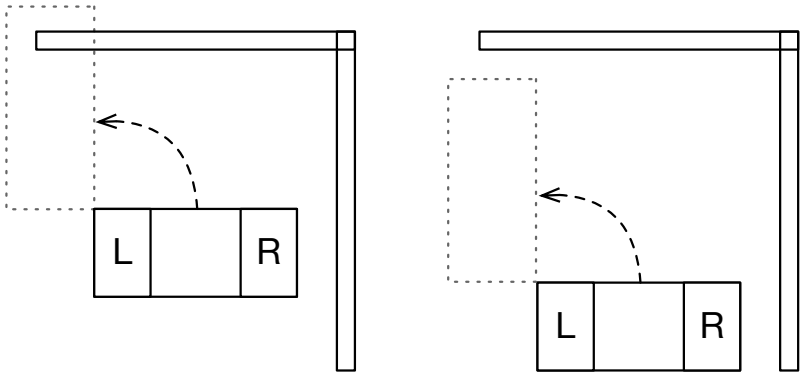


Figura 17: Cas problemàtic

S'hauria d'avaluar quan es disposés d'un dispositiu funcional, però en aquest cas els sensors posteriors serien útils per retrocedir i realitzar una rotació del dispositiu fins que tots els sensors detectessin distàncies segures, i així poder sortir d'aquest cas problemàtic.

Un altre punt a tenir present, és el fet que a causa del gir en forma de compàs que s'ha escollit, la distància que es considera segura hauria de permetre realitzar una rotació sense col·lisionar mentre es realitza la rotació.

En la figura 18a, s'observa com si el dispositiu comença a realitzar la rotació molt tard, col·lisionarà amb l'obstacle. En canvi en la figura 18b es realitza la rotació amb una distància major i pot realitzar la rotació correctament.



(a) Rotació amb una distància no segura

(b) Rotació amb una distància segura

Figura 18: Rotacions amb diferents distàncies segures

Tal com s'ha explicat anteriorment, la distància segura és modificable, i per tant durant les proves reals s'haurà d'observar quina és la distància més recomanable.

Les proves que s'han anat realitzant es basaven generalment en anar apropant un obstacle al sensor i comprovar que els motors tenien el comportament esperat, és a dir, si es seguia l'algorisme correctament.

En les proves la distància que es considerava "segura" era de 10 cm. En el funcionament real, evidentment no es consideraria segura aquesta distància, però per realitzar les proves era més pràctic realitzar-ho amb distàncies curtes.

S'ha realitzat un vídeo per mostrar el funcionament del dispositiu seguint l'algorisme especificat. En la figura 19 es pot observar una captura de pantalla del vídeo. S'han utilitzat ajudes visuals per entendre millor que està succeint en cada instant.

En la part superior esquerra es pot observar l'estat dels sensors, i en la part superior dreta, la direcció que està seguint el dispositiu segons l'estat dels sensors.

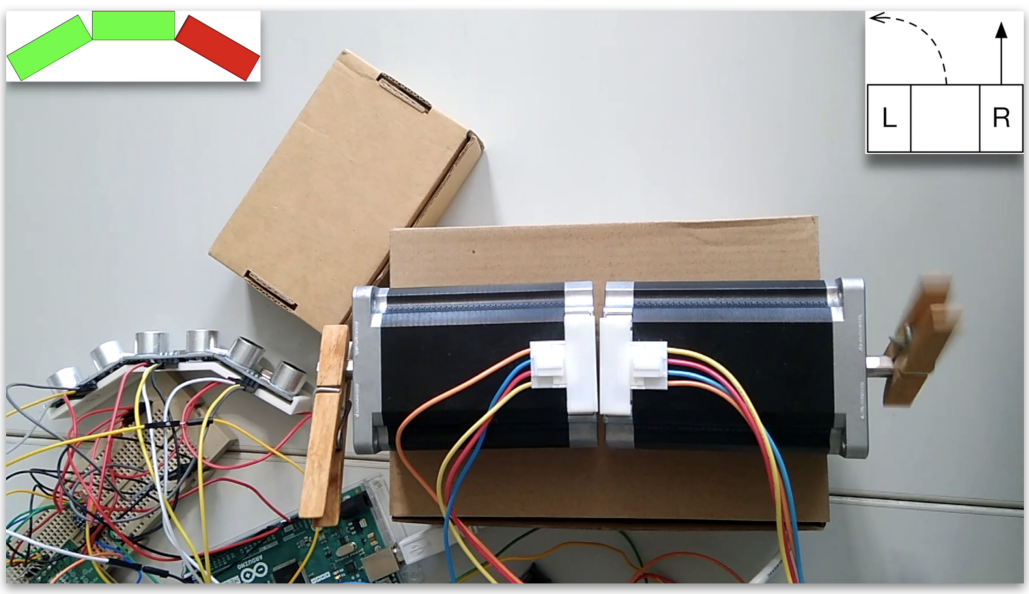


Figura 19: Vídeo del dispositiu seguint l'algorisme de moviment autònom.
Veure vídeo complet en el següent enllaç:
`https://drive.google.com/file/d/1eT5LSN7RH8mzN8LUkZ3jY5313_2wCrYG`

Algunes proves que s'han realitzat per provar el seu funcionament han sigut les següents:

- Comprovar que els diferents casos de l'algorisme realitzen la seva acció corresponent.
- Comprovar que es detecta correctament els obstacles que tenen una inclinació de 30º.
- Modificar el valor de la distància segura i comprovar que el funcionament és correcte.
- Deixar un obstacle fix davant d'un sensor i veure com sempre tria la mateixa direcció.
- Anar apropant progressivament l'obstacle com si el dispositiu s'estigués desplaçant. No es podien moure els sensors, ja que els cables es desconnecten amb facilitat a causa de la seva longitud.
- Comprovar que la velocitat amb la qual canviava de direcció era prou ràpida. Per exemple, deixant caure inesperadament un obstacle davant del sensor o moure l'obstacle entre els sensors laterals.
- Provar com rebotava amb botelles d'aigua. Aquesta prova es va realitzar per curiositat, per observar que succeïa en cas de tenir un obstacle amb una forma circular i transparent. De vegades les mesures no eren del tot exactes, però generalment es detectaven les distàncies correctament.

3.5 Comandament de PS2

El comandament de la consola PlayStation 2 s'utilitza per comunicar-se amb l'Arduino per poder especificar la direcció del dispositiu i així poder controlar-lo de forma manual.

Les direccions s'especifiquen utilitzant la creu direccional del comandament (veure figura 20), ja que ofereix un ús intuïtiu i senzill per poder moure el dispositiu. Quan no hi ha cap botó premut s'indica als motors que s'han d'aturar.

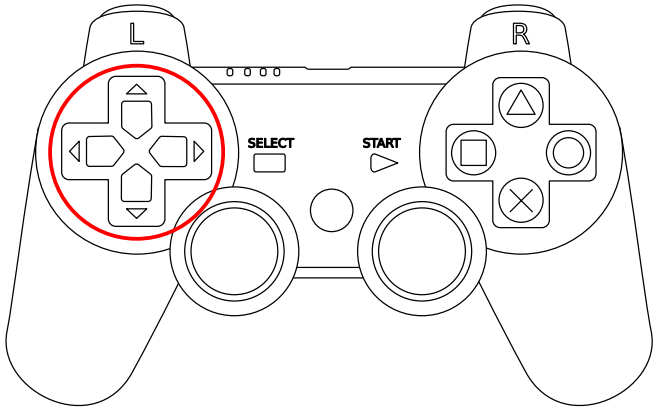


Figura 20: Creu direccional del comandament de PS2

La comunicació es va realitzar mitjançant la llibreria *PS2X*. Amb aquesta llibreria es pot obtenir l'estat dels diferents botons i *joysticks* del comandament. S'explica amb més detall en la secció 3.7.3.

Per no sobrecarregar el processador de l'Arduino, s'ha utilitzat una interrupció per temps que es produeix cada 100ms. En aquesta interrupció s'obté l'estat dels botons i en cas que s'hagi premut un dels botons de la creu direccional, llavors es mou el dispositiu en la direcció especificada, però si es detecta que no se n'ha premut cap llavors s'indica als motors que s'han d'aturar.

En aquest mode de funcionament no s'utilitzen els sensors ultrasònics, però en un futur podria ser útil afegir un mode de funcionament en el qual el dispositiu es controlés mitjançant el comandament, però també utilitzés els sensors per evitar col·lisions.

Per facilitar aquestes ampliacions futures, s'ha utilitzat una interrupció per temps diferent de la que s'utilitza en els sensors ultrasònics, ja que així es poden utilitzar concurrentment les dues interrupcions.

Finalment, també s'ha realitzat un vídeo per mostrar el seu funcionament. Es pot observar una captura de pantalla del vídeo en la figura 21.

S'han afegit ajudes visuals per entendre millor que està succeint en cada instant. En la part superior esquerra s'indica l'estat del comandament, i en la part superior dreta la direcció que segueix el dispositiu.

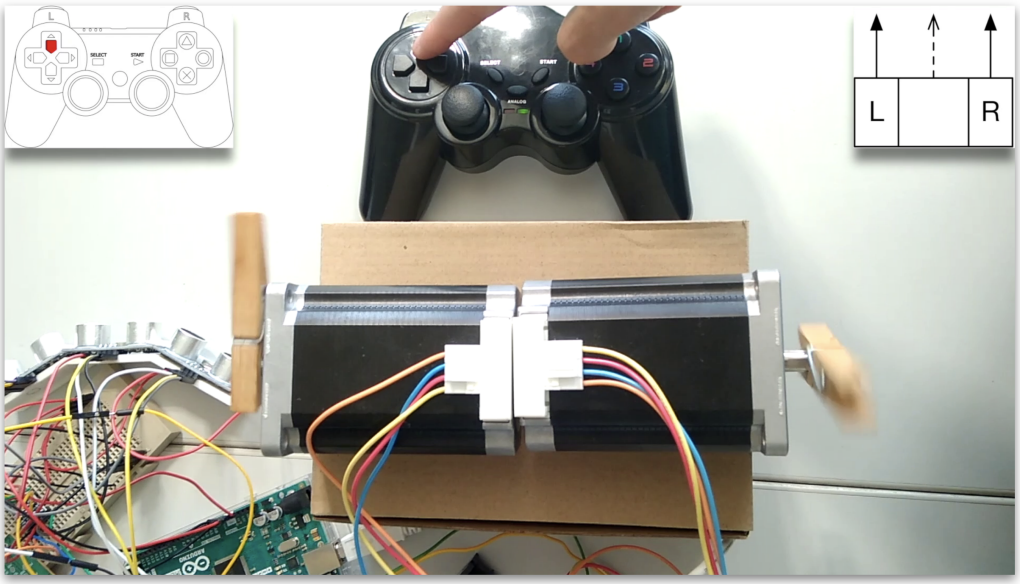


Figura 21: Vídeo del dispositiu controlat amb el comandament de PS2.
Veure vídeo complet en el següent enllaç:
`https://drive.google.com/open?id=1fa0GLZjG9QGC6JyNa5vM9U0PQ9-WcSXD`

3.6 Modes de funcionament

El dispositiu té 3 modes de funcionament, més un mode en el qual és dispositiu està immòbil i es queda esperant que s'activi un dels altres modes de funcionament.

Hi ha dos modes en el qual el dispositiu realitza moviment autònom evitant obstacles. Un d'aquests modes realitza el moviment mentre un botó està premut i l'altre només durant un nombre determinat de segons.

A més d'aquests modes autònoms, també hi ha el mode en el qual el dispositiu es controla manualment amb un comandament de PS2.

Tots els modes segueixen la mateixa estructura per així facilitar l'addició de nous modes de funcionament. Un mode de funcionament té les següents parts:

- Activar: Activar interrupcions i inicialitzar variables.
- Actualitzar direcció: Actualitzar en quina direcció s'ha de moure dispositiu.
- Desactivar: Desactivar interrupcions perquè no interfereixin amb el pròxim mode.

És una estructura senzilla però que permet entendre amb facilitat el funcionament, i permet que l'addició d'un nou mode de funcionament sigui senzilla i no afecti el funcionament dels altres modes.

Cada mode acabarà realitzant les seves operacions concretes, però al final tots acabaran decidint quina direcció ha de prendre el dispositiu.

3.7 Llibreries utilitzades

Durant el desenvolupament s'han utilitzat diverses llibreries per ajudar en determinades funcionalitats. En aquesta secció s'explica perquè s'han utilitzat i quines són les seves característiques més rellevants.

3.7.1 *AccelStepper*

La llibreria *AccelStepper* [9] ha sigut desenvolupada per Mike McCauley i serveix per controlar els motors pas a pas, especialment per poder realitzar acceleracions i desacceleracions. Arduino porta incorporada una llibreria per controlar motors pas a pas, però està orientada per a aplicacions simples, en canvi aquesta llibreria ofereix unes determinades funcionalitats addicionals necessàries per al desenvolupament del dispositiu.

Les funcionalitats més importants d'aquesta llibreria són:

- Acceleració i desacceleració.
- Moure fins a una determinada posició sense bloquejar el funcionament de l'Arduino.
- Poder controlar múltiples motors simultàniament, cadascun amb les seves característiques.

La llibreria disposa de dues llicències. Disposa de la llicència de programari lliure GPL V2, i també disposa d'una llicència comercial per a desenvolupar aplicacions propietàries.

3.7.2 *NewPing*

La llibreria *NewPing* [10], desenvolupada per Tim Eckel, serveix per facilitar la mesura de distàncies amb sensors ultrasònics. Finalment aquesta llibreria no s'ha utilitzat però s'han realitzat diverses proves durant gran part desenvolupament.

En la inicialització, a més d'establir els diferents pins, també s'estableix la distància límit del sensor. En cas que una ona sobrepassi la distància límit, es retorna 0. A continuació s'expliquen les funcionalitats més rellevants:

- Enviar ona i retornar temps d'echo en microsegons. També disposa de funcions per convertir aquest temps en la distància en centímetres i polzades.

- Enviar ona i retornar distància en centímetres o polzades.
- Enviar múltiples ones i retornar la mediana entre els diferents temps.

Tal com s'ha explicat en el desenvolupament, finalment no s'ha pogut utilitzar perquè bloquejava el funcionament de l'Arduino quan s'enviava una ona i els motors pas a pas no podien continuar amb el seu funcionament. A més, la forma d'utilitzar-la de forma concurrent consumia molts recursos de l'Arduino.

L'autor utilitza llicència Copyright, i no permet obres derivades sense el seu permís. Si es vol millorar la llibreria o afegir una nova funcionalitat, s'ha de comunicar a l'autor i ell s'encarregarà de validar-ho.

3.7.3 *PS2X*

La llibreria *PS2X* [11], desenvolupada per Bill Porter sota la llicència GNU GPL, serveix per comunicar l'Arduino amb un comandament analògic de la consola PlayStation 2.

Aquesta llibreria ofereix funcionalitats per poder detectar la pressió que es produeix en els diferents botons de forma analògica, és a dir, es pot detectar amb quanta força s'ha premut un determinat botó. A més dels botons, també permet detectar la posició dels *joysticks*.

Una altra funcionalitat que ofereix és la de poder especificar la vibració i la intensitat d'aquesta.

També permet detectar si els botons han canviat d'estat:

- Botó acaba de ser premut en aquest moment.
- Botó ha canviat d'estat, és a dir, ha passat d'estar premut a no estar-ho, o a l'inrevés.
- Botó acaba de deixar d'estar premut en aquest moment.

3.7.4 *TimerOne* i *TimerThree*

Per realitzar interrupcions per temps és recomanable utilitzar aquestes llibreries com a capa d'abstracció, ja que no és un aspecte trivial perquè s'ha de programar directament els registres de la CPU.

Per realitzar interrupcions per temps s'utilitzen una espècie de rellotges incorporats en l'Arduino que s'anomenen Timers, dels quals n'hi ha diferents tipus:

- Timer 0: 8 bits. S'utilitza en les funcions delay(), millis() i micros(), per tant, és recomanable no modificar-lo.

- Timer 1: 16 bits. S'utilitza en la llibreria Servo de l'Arduino Uno. En el nostre cas no ens afecta, ja que s'utilitza l'Arduino MEGA i l'utilitza en el Timer 5.
- Timer 2: 8 bits. Utilitzat en la funció tone().
- Timer 3, Timer 4 i Timer 5: 16bits. Només disponibles en l'Arduino MEGA.

En el desenvolupament s'ha utilitzat la llibreria *TimerOne* i la *TimerThree* [12], les quals utilitzen els Timers 1 i 3 respectivament.

S'ha utilitzat *TimerOne* per als sensors ultrasònics i *TimerThree* per al comandament de PS2. Tal com s'ha comentat anteriorment, s'ha utilitzat diferents interrupcions per temps per si en un futur s'ha d'utilitzar les dues interrupcions a la vegada.

Aquestes llibreries permeten definir cada quants microsegons s'ha d'executar una determinada funció. També permeten canviar el temps una vegada s'ha inicialitzat, i també permeten activar i desactivar la interrupció.

Actualment la llibreria és mantinguda per Paul Stoffregen sota la llicència CC BY 3.0 US.

3.8 Facilitar la continuació del projecte

Aquest projecte ha de continuar en un futur, per tant és important facilitar el manteniment i treball futur d'aquest projecte.

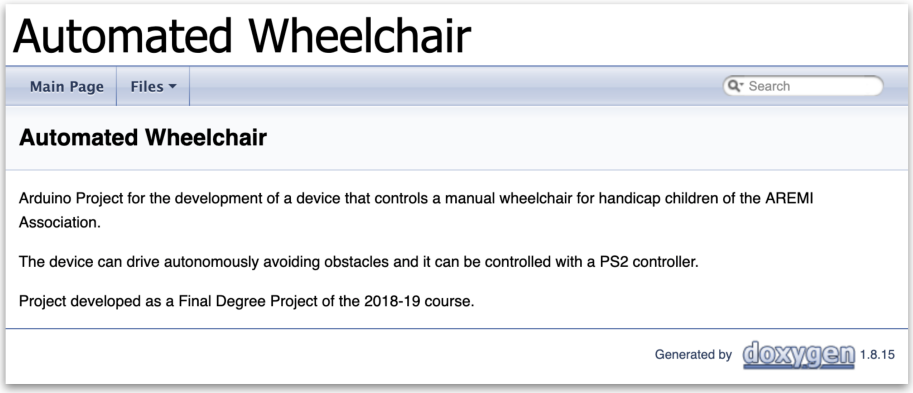
Un aspecte fonamental per facilitar la comprensió del codi és la documentació. S'ha utilitzat l'eina Doxygen per poder generar documentació a partir de comentaris en el codi font. Un dels avantatges d'utilitzar aquesta eina és que, a més de disposar de documentació en el codi font, també es pot consultar en els diferents formats que genera aquesta eina.

La documentació s'ha generat en LaTeX i en HTML. Es pot consultar el resultat de la generació en LaTeX en l'annex A.

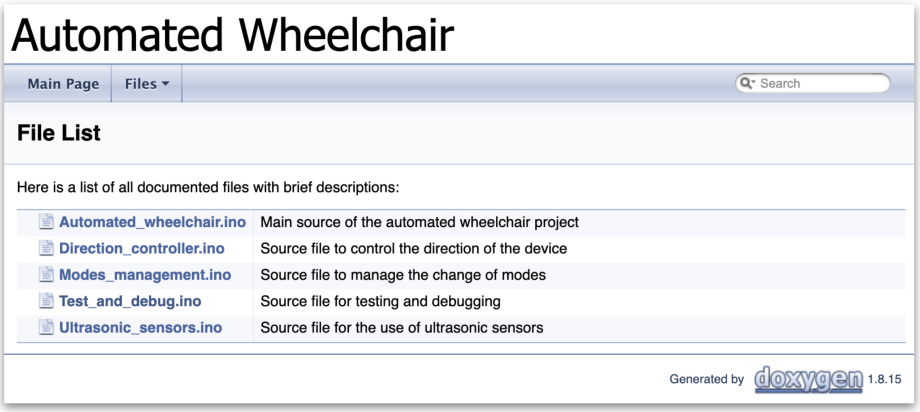
La documentació generada en HTML es pot consultar online. En la figura 22a es pot observar la pàgina principal i en la figura 22b es pot observar la secció *File List* on es disposa dels diferents fitxers del projecte, i es pot consultar la documentació de cadascun d'ells.

Cal remarcar que el codi també conté comentaris que ajuden a millorar la comprensió d'aquest, però que no formen part del document generat, ja que es tracten d'aspectes més específics.

Tot i això, també seria recomanable llegir aquest treball de fi de grau per continuar amb el projecte, ja que així s'entenen millor les decisions de disseny i per quines proves s'ha passat anteriorment per arribar en aquest punt.



(a) Pàgina principal



(b) Secció *File List*

Figura 22: Documentació generada en HTML.
Es pot consultar en el següent enllaç:
`https://ricardzm.github.io/Automated_wheelchair/html/index.html`

Un altre aspecte important és el de facilitar la modificació de variables per poder modificar el funcionament del dispositiu. En l'estat actual, s'han definit uns valors per defecte que s'hauran d'adaptar quan es provi el funcionament real del dispositiu, però actualment són els valors utilitzats per al prototip no funcional. Les variables que es poden modificar fàcilment són les següents:

- Acceleració dels motors pas a pas en passos/segon. Per defecte s'ha definit una acceleració de 100 passos/segon, ja que es considera una acceleració suau.
- Velocitat màxima dels motors pas a pas en passos/segon. Actualment s'ha definit en 300 passos/segon, tot i això quan es disposi d'un prototip funcional, seria recomanable començar a fer les proves amb velocitats baixes, ja que així es poden evitar col·lisions mentre encara s'està ajustant el seu funcionament.

- Distància segura en centímetres per als modes de funcionament on el dispositiu realitza moviment de forma autònoma evitant obstacles. Actualment, la distància està definida en 10 centímetres, però tal com s'ha explicat anteriorment és una distància que només és útil per l'estat actual, ja que quan es disposi d'un prototip funcional s'haurà d'observar quina distància és la més adequada per poder realitzar una rotació sense col·lisionar.
- Nombre de segons que realitza moviment autònom en el mode on només ho realitza durant un nombre determinat de segons. Per defecte es realitza només durant 5 segons, ja que així ho va especificar l'Associació AREMI en el treball de fi de grau anterior.
- Cicles de mesura entre sensors en microsegons. Per defecte està definit en 60000us (60ms).
- Temps entre les comprovacions de l'estat dels botons del comandament de PS2 en microsegons. Per defecte està definit en 100000us (100ms).

Els pins on estan connectats els diferents components no són variables i són constants, ja que no és un valor que s'hagi de modificar durant el funcionament del dispositiu, però si en un futur s'haguessin de modificar per aspectes d'electrònica, també seria senzill modificar-los. En l'annex B es poden consultar les connexions utilitzades.

L'organització del programa s'ha separat en diverses seccions i cadascuna d'elles s'encarrega d'un tema concret. Les seccions són les següents.

- Programa principal.
- Modes de funcionament.
- Controlador de direcció.
- Gestió de sensors ultrasònics.
- *Testing* i *debug*.

En la part de *testing* i *debug*, s'ha de tenir en compte que les tasques de *debug* no es poden realitzar afegint punts d'aturada per comprovar els valors. Per tant, per les tasques de *debug* s'utilitza el port sèrie per enviar i rebre informació. Tot i això, es va definir de tal forma que es poguessin desactivar fàcilment tots els missatges de *debug*, ja que l'Arduino s'atura per enviar el missatge i quan es tracta de molts missatges, pot alentir el seu funcionament.

Per realitzar proves sense dependre dels components físics es van realitzar dos casos de prova en els quals s'enviava un caràcter pel port sèrie per realitzar una tasca determinada:

- Per canviar de mode de funcionament s'introdueix un dels següents caràcters:
  - c: Moviment autònom de forma continuada.

  - t: Moviment autònom durant un temps determinat.
  - p: Controlar el dispositiu amb el comandament de PS2.
  - s: Mode en espera sense realitzar cap acció.
  - u: Mode desconegut, no realitza cap acció i informa que no és un mode vàlid.

- Per moure el dispositiu en una determinada direcció s'utilitzen els següents caràcters:
  - f: Avançar en línia recta.
  - l: Girar a l'esquerra.
  - r: Girar a la dreta.
  - b: Dirigir en dispositiu enrere.
  - u: Direcció desconeguda, no realitza cap acció i informa que no és una direcció vàlida.

Es recomana realitzar les proves de forma aïllada centrant-se en realitzar només una prova a la vegada.

El tema del *testing* en Arduino és força manual, ja que no seria prou fiable utilitzar un emulador per provar el codi, ja que en l'emulador podria funcionar tot correctament però en la realitat el funcionament podria no ser l'esperat per les limitacions dels components físics.

Finalment, es pot consultar el codi en el següent enllaç. També es disposa del fitxer de configuració de Doxygen per generar la documentació.

`https://github.com/ricardzm/Automated_wheelchair`

4 Treball futur

Tal com s'ha anat comentant, es tracta d'un projecte que tindrà una gran part de treball futur i s'ha tingut present durant tot el desenvolupament que aquest projecte l'ha de continuar una altra persona, i per tant s'ha de facilitar la seva continuació.

Per fer-ho s'ha realitzat pensat, què m'agradaria trobar si jo hagués de continuar un projecte que hagués començat una altra persona. En aquest cas, m'agradaria disposar d'una documentació coherent, ja que ajuda a entendre que fa una determinada funció sense haver d'esbrinar que realitza el codi que la comprèn.

Un altre aspecte és que contingui una estructura coherent, ja que si no també dificulta el seguiment i la comprensió del codi.

Sobretot, s'ha orientat a facilitar la continuació per a la pròxima persona. S'ha desenvolupat de forma que sigui senzill modificar el funcionament amb variables, afegir nous modes de funcionament, poder utilitzar els diferents components sense interferències entre ells...

El treball futur d'aquest projecte pot ser de diferents rames:

- Interacció persona ordinador.
- Electrònica i mecànica.
- Intel·ligència artificial.

Tot i això, la continuació del projecte hauria de començar per tenir l'estructura del dispositiu i poder provar el seu funcionament, ja que no tindria sentit avançar en noves funcionalitats, si encara no s'ha pogut provar que el funcionament bàsic sigui prou bo perquè sigui utilitzat pels seus usuaris.

En les proves del moviment autònom s'hauria de tenir present la possibilitat d'afegir sensors posteriors per poder moure el dispositiu enrere. En l'estat actual només es disposa de sensors frontals, i per tant no es retrocedeix mai, ja que el dispositiu desconeix els obstacles hi poden haver en la part posterior. Retrocedir podria ser útil per evitar casos on avançar o rotar no seria prou segur a causa de la distància dels obstacles en la part frontal.

Quan el dispositiu tingués un funcionament prou fiable, s'hauria de portar a l'Associació AREMI i ensenyar el funcionament del dispositiu. En aquesta visita es comentaria amb l'associació quines millores s'haurien de realitzar, i quines funcionalitats necessitarien per la pròxima versió.

Un dels punts que es va demanar per part de l'associació en l'anterior treball de fi de grau, va ser poder realitzar recorreguts pel recinte. Aquest punt s'hauria d'explorar amb més detenciment com es podria realitzar, però durant el desenvolupament es va parlar de la possibilitat d'utilitzar unes balises de localització en interiors per poder detectar la ubicació del dispositiu i així dirigir-lo cap a una zona del recinte.

També seria interessant utilitzar aquests dispositius per canviar el funcionament segons en quina zona està ubicat el dispositiu. Per exemple, si el dispositiu està en una zona sense obstacles, podria tenir una velocitat major.

Durant el desenvolupament s'ha parlat de la necessitat d'una interfície gràfica per poder configurar el dispositiu i obtenir informació sobre el seu funcionament.

La interfície podria estar integrada en el mateix dispositiu, però també es podria disposar d'una aplicació mòbil amb la qual es pogués configurar, obtenir informació i fins i tot controlar el moviment del dispositiu.

En aquesta interfície també seria interessant poder tenir diferents configuracions pels diferents tipus de cadires de rodes. L'estructura del dispositiu està dissenyada per ser utilitzada en diferents cadires de rodes, però també es podria adaptar el funcionament segons el tipus de cadira que s'està utilitzant.

Quan es disposés d'un dispositiu amb un bon funcionament es podrien explorar altres aspectes d'intel·ligència artificial, per desenvolupar algorismes de moviment autònom segons la ubicació del dispositiu en el recinte.

Un altre aspecte a explorar seria la possibilitat de combinar-ho amb altres projectes del grup de recerca GRIHO, ja que tal com es pot observar, és un projecte amb moltes possibilitats.

5 Conclusions

Els objectius del treball s'han pogut complir, tot i això, el fet de no disposar d'un prototip funcional del dispositiu ha fet que no es pogués provar el seu funcionament real. A més, tampoc s'ha pogut portar a l'Associació AREMI per mostrar el seu funcionament.

S'ha pogut implementar la lògica necessària per poder utilitzar el dispositiu amb els seus components, procurant sempre que l'ús concurrent de diferents components no afectés el funcionament individual de cadascun d'ells.

També s'han pogut desenvolupar els diferents modes de funcionament, sent el moviment autònom el més complicat pel fet d'haver de combinar el funcionament dels motors pas a pas i els sensors ultrasònics.

Tal com s'ha comentat anteriorment un dels objectius també era facilitar la continuació, ja que es tracta d'un projecte que s'anirà expandint en un futur, i per tant s'ha seguit una estructura coherent i documentat per facilitar la seva comprensió.

Aquest treball també ha servit per poder entendre millor com funciona un microcontrolador, ja que s'ha de desenvolupar a tan baix nivell que no es disposa de funcionalitats com per exemple fils d'execució.

Ha sigut interessant també el fet de treballar en un projecte multidisciplinari, ja que s'havia de contemplar aspectes informàtics, mecànics i electrònics. Es va haver de començar de zero, perquè no es coneixia la plataforma Arduino ni cap plataforma semblant. A més, els coneixements sobre electrònica o mecànica que es tenien eren molt bàsics.

Personalment, no havia treballat mai en cap plataforma d'aquest tipus i ha sigut interessant veure com s'ha de desenvolupar pensant en el component físic i les seves limitacions, ja que he vist com s'ha de comprendre el seu funcionament a baix nivell abans de poder desenvolupar una funcionalitat amb aquell component.

Espero que el grup de recerca GRIHO, pugui continuar amb el projecte perquè finalment l'Associació AREMI pugui disposar d'un dispositiu que estigui adaptat a les necessitats dels seus usuaris, i que es vagi ampliant i millorant amb cada persona que vagi formant part del projecte.

En conclusió, tot i les dificultats, ha sigut un treball interessant, ja que s'ha pogut treballar en una plataforma on la forma de desenvolupar és diferent de les que havia utilitzat fins al moment, i s'ha pogut aprofundir en el funcionament d'un dispositiu en el qual es treballa a molt baix nivell.

6 Bibliografia

Referències

[1] Roger Romagosa Vallès. "Disseny de motorització per cadira de rodes manual per la Fundació Aremi". Grau en Enginyeria Mecànica. Escola Politècnica Superior, Universitat de Lleida.

[2] *BodyViz*. https://www.bodyviz.com/.

[3] Peter Ray Allison. *BBC - Future - The surprising uses of games controllers*. http://www.bbc.com/future/story/20141212-press-x-press-y-fire-laser.

[4] *Doxygen*. http://www.doxygen.nl/.

[5] *Documenting the code*. http://www.doxygen.nl/manual/docblocks.html.

[6] Enrico Formenti. *HC-SR04: using multiple ultrasonic modules*. https://macduino.blogspot.com/2013/11/hc-sr04-using-multiple-ultrasonic.html.

[7] Tim Eckel. *NewPing Arduino Library for Arduino*. https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home#!examples.

[8] *Ultrasonic Ranging Module HC - SR04*. https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf.

[9] Mike McCauley. *AccelStepper library for Arduino*. http://www.airspayce.com/mikem/arduino/AccelStepper/.

[10] Tim Eckel. *NewPing Library for Arduino*. https://playground.arduino.cc/Code/NewPing.

[11] Bill Porter. *PlayStation 2 Controller Arduino Library v1.0*. http://www.billporter.info/2010/06/05/playstation-2-controller-arduino-library-v1-0/.

[12] *TimerOne & TimerThree Arduino Libraries*. https://www.pjrc.com/teensy/td_libs_TimerOne.html.

[13] *Controlando el rover con un mando PS2*. https://www.prometec.net/controlador-ps2/.

[14] Lluis Llamas. *Controla arduino con el mando inalámbrico de la PS2*. https://www.luisllamas.es/controla-arduino-con-el-mando-inalambrico-de-la-ps2/.

[15] Enrico Formenti. *HC-SR04: ultrasonic sensor for Arduino*. https://macduino.blogspot.com/2013/11/HC-SR04-part1.html.

[16] *¿Qué es un sensor ultrasónico?* https://www.keyence.com.mx/ss/products/sensor/sensorbasics/ultrasonic/info/index.jsp.

[17] James Lewis. *Picking the Right Arduino*. https://blog.hackster.io/picking-the-right-arduino-341a0a9550c7.

7 Annexos

Índex d'annexos

Automated Wheelchair

Generated by Doxygen 1.8.15

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 Automated_wheelchair.ino File Reference

Main source of the automated wheelchair project.

```
#include <AccelStepper.h>
#include <TimerOne.h>
#include <TimerThree.h>
#include <PS2X_lib.h>
```

Macros

- #define **DEBUG**
- #define **DEBUG_PRINT**(x) Serial.println(x)
- #define **RIGHT_STEPPER_STEP_PIN** 22
- #define **RIGHT_STEPPER_DIRECTION_PIN** 23
- #define **LEFT_STEPPER_STEP_PIN** 24
- #define **LEFT_STEPPER_DIRECTION_PIN** 25
- #define **LEFT_ECHO_PIN** 2
- #define **LEFT_TRIGGER_PIN** 8
- #define **CENTER_ECHO_PIN** 3
- #define **CENTER_TRIGGER_PIN** 9
- #define **RIGHT_ECHO_PIN** 20
- #define **RIGHT_TRIGGER_PIN** 10
- #define **PS2_CLOCK_PIN** 14
- #define **PS2_COMMAND_PIN** 15
- #define **PS2_ATTENTION_PIN** 16
- #define **PS2_DATA_PIN** 17
- #define **LEFT_SENSOR** 0
- #define **CENTER_SENSOR** 1
- #define **RIGHT_SENSOR** 2
- #define **FORWARD** 0
- #define **BACKWARD** 1
- #define **LEFT** 2
- #define **RIGHT** 3
- #define **TEMPORAL_SELF_DRIVING** 0
- #define **CONTINUOUS_SELF_DRIVING** 1
- #define **PS2_CONTROLLER** 2
- #define **STAND_BY** 3

Functions

- AccelStepper **right_stepper** (AccelStepper::DRIVER, RIGHT_STEPPER_STEP_PIN, RIGHT_STEPPER↩ _DIRECTION_PIN)
- AccelStepper **left_stepper** (AccelStepper::DRIVER, LEFT_STEPPER_STEP_PIN, LEFT_STEPPER_DI↩ RECTION_PIN)
- void **setup** ()
- void **loop** ()

Variables

- int **current_mode** = STAND_BY
- long **sensors_period** = 60000
- long **PS2_period** = 100000
- int **safe_distance** = 10
- int **max_speed** = 300
- int **acceleration** = 100
- int **self_driving_seconds** = 5
- unsigned long **end_self_driving**
- volatile unsigned long **sensors_times** [3][2]
- volatile int **current_sensor** = 0
- volatile bool **one_cycle** = false
- volatile unsigned long **last_time** = 0
- PS2X **ps2x**

2.1.1 Detailed Description

Main source of the automated wheelchair project.

Author

Ricard Zapater Muñoz

Version

1.0

Source file with the setup and the loop functions. In this file there are also all the configurable variables, pins and constants.

2.2 Direction_controller.ino File Reference

Source file to control the direction of the device.

Functions

- void update_direction ()

  Update the direction of the device depending on the current mode.
- void update_temporal_self_driving_direction ()

  Update the direction of the temporal self-driving mode. Disable the mode if the time has passed.
- void update_ps2_controller_direction ()

  Read the state of the PS2 controller and move the device to the direction specified with the controller.
- void update_self_driving_direction ()

  Move the device to the safest direction.
- void move_direction (int dir)

  Set steps to move the device to a direction.

2.2.1 Detailed Description

Source file to control the direction of the device.

Author

Ricard Zapater Muñoz

Version

1.0

2.2.2 Function Documentation

2.2.2.1 update_direction()

```
void update_direction ( )
```

Update the direction of the device depending on the current mode.

2.2.2.2 update_temporal_self_driving_direction()

```
void update_temporal_self_driving_direction ( )
```

Update the direction of the temporal self-driving mode. Disable the mode if the time has passed.

2.2.2.3 update_ps2_controller_direction()

```
void update_ps2_controller_direction ( )
```

Read the state of the PS2 controller and move the device to the direction specified with the controller.

To control the direction of the device use the Directional Pad. If no button is pressed, then the steppers stop.

2.2.2.4 update_self_driving_direction()

```
void update_self_driving_direction ( )
```

Move the device to the safest direction.

2.2.2.5 move_direction()

```
void move_direction (
            int dir )
```

Set steps to move the device to a direction.

Parameters

dir	Direction to move the device. Use the defined directions in Automated_wheelchair.ino

2.3 Modes_management.ino File Reference

Source file to manage the change of modes.

Functions

- void change_current_mode (int mode)
  Disable the current mode and enable the specified mode. Stop the steppers if they are running.
- void enable_mode (int mode)
  Enable the specified mode.
- void enable_self_driving ()
  Setup the interruptions and pins to enable self-driving.
- void enable_ps2_controller ()
  Setup the interruptions to use the controller.
- void disable_mode (int mode)
  Disable the specified mode.
- void disable_self_driving ()
  Disable self-driving by detaching the interrupt.
- void disable_PS2_controller ()
  Disable the use of the controller by detaching the interrupt.

2.3.1 Detailed Description

Source file to manage the change of modes.

Author

Ricard Zapater Muñoz

Version

1.0

Source file to change modes. All the modes must have an enable function and a disable function. Use the defined modes in Automated_wheelchair.ino

2.3.2 Function Documentation

2.3.2.1 change_current_mode()

```
void change_current_mode (
            int mode )
```

Disable the current mode and enable the specified mode. Stop the steppers if they are running.

Parameters

mode	Change current mode to this mode.

2.3.2.2 enable_mode()

```
void enable_mode (
            int mode )
```

Enable the specified mode.

Parameters

mode	Mode to enable.

2.3.2.3 enable_self_driving()

```
void enable_self_driving ( )
```

Setup the interruptions and pins to enable self-driving.

2.3.2.4 enable_ps2_controller()

```
void enable_ps2_controller ( )
```

Setup the interruptions to use the controller.

2.3.2.5 disable_mode()

```
void disable_mode (
            int mode )
```

Disable the specified mode.

Parameters

mode	Mode to disable.

2.3.2.6 disable_self_driving()

```
void disable_self_driving ( )
```

Disable self-driving by detaching the interrupt.

2.3.2.7 disable_PS2_controller()

```
void disable_PS2_controller ( )
```

Disable the use of the controller by detaching the interrupt.

2.4 Test_and_debug.ino File Reference

Source file for testing and debugging.

Functions

- void test_change_mode ()

  Test the change of mode by writing a character in the serial port.
- String mode_to_string (int mode)

  Get String of a mode.
- void debug_stepper_info (AccelStepper stepper)

  Print debug information of a stepper. Information for debugging:
- void test_directions ()

  Test the directions by writing a character in the serial port.

2.4.1 Detailed Description

Source file for testing and debugging.

Author

Ricard Zapater Muñoz

Version

1.0

Source file with tests and functions to help debugging.

2.4.2 Function Documentation

2.4.2.1 test_change_mode()

```
void test_change_mode ( )
```

Test the change of mode by writing a character in the serial port.

Characters for changing the mode:

- t: Temporal self-driving mode
- c: Continuous self-driving mode
- p: PS2 controller mode
- s: Stand by mode
- u: Unknown mode

2.4.2.2 mode_to_string()

```
String mode_to_string (
            int mode )
```

Get String of a mode.

Parameters

mode	Mode to obtain its string

Returns

String Mode in string

2.4.2.3 debug_stepper_info()

```
void debug_stepper_info (
            AccelStepper stepper )
```

Print debug information of a stepper. Information for debugging:

- Current position
- Target position
- Distance to go
- Speed

Parameters

stepper	Stepper to obtain the debug information

2.4.2.4 test_directions()

```
void test_directions ( )
```

Test the directions by writing a character in the serial port.

Characters for the directions:

- f: Forward
- b: Backward
- l: Left
- r: Right
- u: Unknown direction

2.5 Ultrasonic_sensors.ino File Reference

Source file for the use of ultrasonic sensors.

Functions

- void calculate_sensors_distances (double distances[ ])

  Calculate de sensors distances given the emission time and the reception time.
- int get_safe_direction (double distances[ ])

  Return the safest direction analyzing the distances from the sensors.
- void check_sensor_echo_signal (int num_sensor, int echo_pin)

  Check the pin signal of the sensor's echo.
- void echo_interrupt_left_sensor ()

  Function to call when the echo pin of the left sensor produces an interruption. (Functions called with an interruption can't receive parameters)
- void echo_interrupt_center_sensor ()

  Function to call when the echo pin of the center sensor produces an interruption. (Functions called with an interruption can't receive parameters)
- void echo_interrupt_right_sensor ()

  Function to call when the echo pin of the right sensor produces an interruption. (Functions called with an interruption can't receive parameters)
- void trigger_current_sensor ()

  Send the trigger for the current sensor.

2.5.1 Detailed Description

Source file for the use of ultrasonic sensors.

Author

Ricard Zapater Muñoz

Version

1.0

2.5.2 Function Documentation

2.5.2.1 calculate_sensors_distances()

```
void calculate_sensors_distances (
            double distances[] )
```

Calculate de sensors distances given the emission time and the reception time.

Parameters

distances	Array to overwrite with the distances

2.5.2.2 get_safe_direction()

```
int get_safe_direction (
            double distances[] )
```

Return the safest direction analyzing the distances from the sensors.

Parameters

distances	Distances from the sensors.

Returns

int Safest direction.

2.5.2.3 check_sensor_echo_signal()

```
void check_sensor_echo_signal (
            int num_sensor,
            int echo_pin )
```

Check the pin signal of the sensor's echo.

If the signal is HIGH it's the start, if it's LOW it's the end.

Parameters

num_sensor	Number of the sensor.
echo_pin	Sensor's echo pin.

2.5.2.4 trigger_current_sensor()

```
void trigger_current_sensor ( )
```

Send the trigger for the current sensor.

Annex B - Connexions

Motor pas a pas dretà:

- Step: 22
- Direction: 23

Motor pas a pas esquerrà:

- Step: 24
- Direction: 25

Sensor ultrasònic esquerrà:

- Echo: 2
- Trigger: 8

Sensor ultrasònic central:

- Echo: 3
- Trigger: 9

Sensor ultrasònic dretà:

- Echo: 20
- Trigger: 10

Comandament de PS2:

- Clock: 14
- Command: 15
- Attention: 16
- Data: 17

Consideracions

En els MicroStep Driver dels motors pas a pas, hi ha la connexió ENA+(+5V) que no està connectada.

En l'Arduino MEGA els pins en els quals es poden realitzar interrupcions són els pins: 2, 3, 18, 19, 20 i 21. En un futur és possible que es necessiti algun pin d'aquests per poder realitzar una interrupció, per tant els pins que es podrien aprofitar serien: 18, 19 i 21, ja que els altres s'estan utilitzant per als sensors ultrasònics.